UNIVERSITÉ DE MONTRÉAL

**CENTRE DE RECHERCHE SUR LES TRANSPORTS**

MÉMOIRE D'ÉTUDIANT

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

# L'ALLOCATION DES APPAREILS POUR UN HORAIRE AÉRIEN : UNE APPROCHE INTERACTIVE-GRAPHIQUE

par

Denis Lebeuf

Mémoire de M.Sc., département d'informatique et de recherche opérationnelle, dirigé par Michael Florian

Université de Montréal

Centre de recherche sur les transports - Publication #415

Avril 1985

Dépôt légal - 3e trimestre 1985

## TABLE DES MATIERES


LISTE DES FIGURES v

LISTE DES TABLEAUX v

SOMMAIRE vi

1. INTRODUCTION 1

2. SITUATION DU PROBLEME 5

   2.1 Présentation du problème 5
   2.2 Solution envisagée 10

3. ANALYSE ET MODELISATION DU PROBLEME 18

   3.1 Quasi-périodicité des données 18
   3.2 Definition des termes et notation 21
   3.3 Opérateurs et propriétés de base 26
   3.4 Génération des accrochages 32

4. ASPECTS INFORMATIQUES 39

   4.1 Implantation des vecteure de bits 39
   4.2 Organisation logique de la base de données 47
   4.3 Structures de données 57

5. REPRESENTATION GRAPHIQUE DU RESEAU AERIEN 77

6. INTERFACE ENTRE L'USAGER ET LE SYSTEME 85

7. LES MODULES 92

7.1 Les utilitaires 94
7.2 Les données de base 96
7.3 Les activités 97
7.4 Les accrochages 102
7.5 Les tâches des véhicules 111

8. CONCLUSION 113

APPENDICE A - Fichiers internes de la banque de données de PLANAIR 116

REFERENCES 127

REMERCIEMENTS 129

## LISTE DES FIGURES


| Figure: | | Page: |
|---|---|---|
| 2-1 | Reproduction d'un graphique d'allocation | 8 |
| 2-2 | Illustration d'un cas d'exception | 9 |
| 4-1 | Contenu de la banque de données de PLANAIR | 58 |
| 4-2 | Structure des pointeurs des activités | 63 |
| 4-3 | Structure des pointeurs des accrochages | 67 |
| 4-4 | Listes créées au cours de la phase I de l'algorithme de génération des accrochages | 70 |
| 4-5 | Résultats obtenus avec l'algorithme de génération des accrochages | 76 |
| 5-1 | Projection conique polaire | 79 |
| 5-2 | Projection de Mercator | 80 |
| 5-3 | Hierarchie des routines de projections géographiques | 81 |
| 7-1 | Réseau d'accès direct aux aéroports de Dorval et Mirabel | 98 |
| 7-2 | Diagramme espace-temps - (L1011) | 103 |
| 7-3 | Diagramme espace-temps - (B747 ) | 104 |
| 7-4 | Diagramme espace-temps "déplié" | 106 |
| 7-5 | Diagramme appareils-temps prévu | 112 |


## LISTE DES TABLEAUX


| Tableau: | | Page: |
|---|---|---|
| 7-1 | Les modules de PLANAIR | 93 |
| 7-2 | Exemple d'un rapport d'activités | 101 |

## SOMMAIRE

Ce mémoire porte sur l'allocation des appareils d'une flotte aérienne, étant donné un horaire de vols à desservir ("aircraft scheduling"). Pour la majorité des sociétés aériennes, ce problème est résolu en produisant, à la main, des graphiques d'allocation ("routing charts") sur de grands tableaux ou de grandes feuilles de papier.

Nous avons formalisé le problème d'allocation et développé une notation ensembliste permettant d'en modéliser les composantes. Nous introduisons une méthode basée sur une représentation binaire de l'horaire de chaque vol, chaque horaire étant représenté dans son entier de façon très compacte. Cette représentation nous a permis d'exploiter la *quasi-périodicité* sous-jacente des données et de formuler un algorithme d'allocation de type LIFO très efficace, basé sur des opérations logiques parallèles.

Nous décrivons ensuite l'approche et les concepts de base utilisés pour le développement du système interactif-graphique PLANAIR. Il s'agit d'un logiciel intégré de conception assistée par ordinateur (CAO), où l'expérience du planificateur est combinée avec les possibilités graphiques et analytiques de l'ordinateur, en vue de permettre une véritable optimisation interactive. La banque de données du système PLANAIR ainsi que tous les modules y donnant accès ont été implantés puis testés à partir de données fournies par la société Air Canada et provenant d'une situation

réelle.

Nous abordons aussi les problèmes liés à la représentation graphique du réseau aérien sous forme de cartes géographiques et montrons comment ils ont été résolus en définissant avec précaution les routines graphiques de base.

# 1. INTRODUCTION

Pour la plupart des sociétés aériennes, l'allocation des appareils de leur flotte aux vols prévus à l'horaire ("aircraft scheduling") est encore une tâche exécutée manuellement avec le crayon et la table à dessin comme outils de base. On a longtemps cru pouvoir automatiser complètement cette tâche à l'aide de techniques d'optimisation appropriées, mais la grande quantité de contraintes qualitatives et la difficulté d'énoncer une fonction économique valable ont eu raison, jusqu'ici, de cette approche. De grands progrès ont tout de même été réalisés et ont permis une meilleure formalisation et une plus grande compréhension du processus de planification. Cependant, l'habileté d'un planificateur expérimenté est restée le moyen de résolution le plus efficace.

Parallèlement, l'avènement de la micro-informatique a permis de diminuer grandement les coûts d'utilisation de l'ordinateur et de le rendre accessible à un plus grand nombre d'usagers. De plus, les récents développements dans le domaine de la technologie des équipements périphériques a entraîné une remise en question du rôle traditionnellement attribué à l'ordinateur: doté d'une plus grande mémoire et couplé à des équipements graphiques, l'ordinateur devient un outil polyvalent et n'est plus une simple "machine à broyer des nombres".

Ce travail s'inscrit dans le cadre d'une démarche visant à intégrer les deux aspects mathématique et informatique du problème. Il a pour but,

suite à une modélisation adéquate des principaux éléments du problème, de fournir au planificateur aérien un ensemble d'outils interactifs-graphiques qui l'aideront dans la recherche d'une solution optimale. Nous ne cherchons pas, ici, à automatiser le processus de planification; ce serait là négliger l'expérience et la grande connaissance du système en cause, que possède le planificateur, et qui constituent des éléments de solution importants. Nous visons plutôt à développer un logiciel intégré de conception assistée par ordinateur (CAO) qui permette une véritable optimisation interactive.

Plusieurs travaux similaires ont déjà été réalisés par différentes sociétés aériennes (voir, par exemple, Barnard, 1980; Bird et McCallum, 1971; IBM, 1980; Kayukawa, 1983). Ces travaux sont cependant considérés comme relevant des affaires internes de la compagnie. Conséquemment, la littérature disponible reste vague, et ne traite que de l'approche générale utilisée, sans vraiment entrer dans les détails. Les aspects techniques abordés sont souvent très spécifiques à la compagnie concernée et ne peuvent être étendus à un contexte plus général. Il ressort quand même un point important à propos de l'approche générale utilisée: la plupart des logiciels développés jusqu'à maintenant sont constitués d'un modèle d'optimisation autour duquel on a ajouté quelques programmes interactifs. L'intervention humaine est donc reléguée au second plan et se limite souvent à prendre la relève lorsque le modèle mathématique est en difficulté. Nous sommes d'avis que le problème devrait être abordé dans l'autre sens: il faut d'abord développer un ensemble d'outils interactifs-graphiques à partir d'un modèle analytique de base, puis y greffer un ensemble d'outils d'optimisation. Nous voulons que le planificateur ait le plus grand contrôle possible et que ce soit lui qui puisse décider quand,

et dans quel contexte, il a besoin d'utiliser un outil d'optimisation .

Au chapitre deux, nous situerons le problème d'allocation dans son contexte, et décrirons les éléments de solution proposés. Nous y indiquerons de plus le plan de développement du logiciel interactif-graphique PLANAIR.

Par la suite, au chapitre trois, nous détaillerons le modèle analytique développé. Les fondements de ce modèle reposent sur la nature quasi-périodique des horaires des vols. Les horaires présentent en effet une répétition dans le temps qu'il serait avantageux d'exploiter, mais comportent de nombreuses exceptions. Ces exceptions ont été jusqu'ici un obstacle empêchant un traitement systématique et efficace des données. Le modèle que nous proposons tire avantage de la périodicité sous-jacente des données, tout en tenant compte des cas d'exception de façon implicite, et permet un traitement efficace basé sur des opérations logiques parallèles.

Au chapitre quatre, nous discuterons des aspects informatiques liés à l'implantation des concepts développés au chapitre précédent. Nous donnerons le détail des structures de données utilisées et de l'organisation logique de la base de données du logiciel PLANAIR.

Nous traiterons, au chapitre cinq, de la représentation graphique du réseau aérien desservi. Certaines caractéristiques du réseau peuvent être visualisées à l'aide de cartes géographiques. Cette approche pose cependant le problème du choix de la projection géographique utilisée. Nous décrirons les difficultés inhérentes à ce problème et comment il a été résolu.

Les chapitres six et sept, enfin, feront le lien entre les notions théoriques développées et leur application pratique. Nous montrerons comment se présente PLANAIR du point de vue d'un usager, en donnant un aperçu général des modules développés et de leur fonctionnement.

## 2. SITUATION DU PROBLEME

Au cours de ce chapitre, nous exposerons le problème d'allocation que nous a présenté Air Canada et décrirons la méthode actuellement utilisée par cette compagnie pour y faire face. Par la suite, les éléments de solution que nous proposons seront décrits et situés dans le cadre du plan de développement du logiciel baptisé PLANAIR.

### 2.1 Présentation du problème

La planification des opérations d'une compagnie aérienne d'envergure internationale est un problème complexe qui comporte de nombreuses facettes inter-reliées. Par exemple, un département de marketing doit prévoir de nouveaux horaires de vols et de nouveaux services pour adapter le produit offert aux conditions du marché dans un domaine où la concurrence est très forte, tandis qu'un autre département doit, sur la base de ces changements, veiller à une utilisation optimale de la flotte d'appareils pour réaliser ces nouveaux horaires. Parallèlement, il faut planifier en conséquence les horaires des équipages et des équipes de travail au sol (entretien, manutention des bagages et des marchandises, etc.).

Dans ce domaine, les planificateurs doivent faire face à une série de contraintes très strictes concernant l'entretien et le bon fonctionnement de chaque avion. Avant un certain nombre d'heures de vol critique, chacun des avions doit être revenu à une base où un entretien préventif pourra être effectué. A ces règles de sécurité s'ajoutent des contraintes d'ordre

financier: étant donné l'ampleur des coûts d'opération impliqués, la flotte doit être utilisée à son maximum. Les temps morts pendant lesquels un avion est laissé inactif à un aéroport en attente d'un vol doivent être autant que possible évités, tout en tenant compte, cependant, du temps requis pour ravitailler et préparer un appareil pour un nouveau vol. Il faut de plus tenir compte des caractéristiques techniques des différents types d'avions (capacité, rayon d'action, etc.) et du service offert à la clientèle (il faut, entre autres, minimiser le nombre de transbordements à effectuer entre les origines et les destinations). Ce n'est là qu'une énumération brève et succincte des différents problèmes auxquels doivent faire face les planificateurs lors de l'allocation des appareils aux vols d'un horaire, mais qui en montre bien la complexité.

C'est à ce dernier aspect du problème que nous nous intéresserons dans le cadre de ce travail: étant donné un horaire de vols à desservir, comment gérer, à court et moyen termes, la flotte d'avions disponibles de façon efficace. Actuellement, la compagnie Air Canada utilise une méthode manuelle à étapes successives pour planifier, de deux à trois ans d'avance, la composition de sa flotte et l'utilisation de ses avions, étant donné l'horaire des vols qu'il est prévu d'effectuer. Dans un premier temps, le département de marketing détermine un nouvel horaire basé sur les conditions anticipées du marché. Puis un planificateur ("scheduler" en anglais) reprend l'allocation de l'année précédente et y apporte des modifications locales de façon à l'adapter à ces directives. (Présentement, les vols changeant relativement peu d'année en année, il est toujours possible d'adapter ainsi l'allocation de l'année précédente.) Pour chaque type d'appareils, des graphiques des nouvelles allocations ("routing maps") sont alors produits manuellement et envoyés successivement aux différents

départements concernés (marketing, entretien, service du personnel, etc.). Ces derniers indiquent les situations de conflit engendrées par ces allocations et retournent les graphiques au planificateur qui apportera les corrections nécessaires et en produira de nouveaux. Et le cycle recommence. C'est un processus fastidieux qui demande de deux à trois mois. Une partie d'un de ces graphiques est reproduite à la figure 2-1.

Cette méthode comporte plusieurs désavantages:

- Dans certains cas, les graphiques produits sont compliqués et difficiles à bien comprendre. Plusieurs membres du personnel des différents départements qui ont à travailler avec ces graphiques se plaignent de la difficulté d'interprétation. Le problème survient au niveau du traitement des *cas d'exception.* Les cas d'exception sont, par exemple, des vols qui sont ajoutés ou retranchés de l'horaire pour répondre à certaines conditions (haute ou basse saison, congés fériés, etc.) et qui viennent rompre le cycle normal de l'horaire d'un vol. Loin d'être des cas isolés (malgré le vocable "d'exception"), ces cas constituent une partie importante de l'horaire. Sur les graphiques, ils sont indiqués par des annotations qui donnent les dates où ils surviennent et les différentes actions prévues (voir figure 2-2). Ces annotations deviennent souvent cryptiques pour toute autre personne que le planificateur les ayant transcrites.

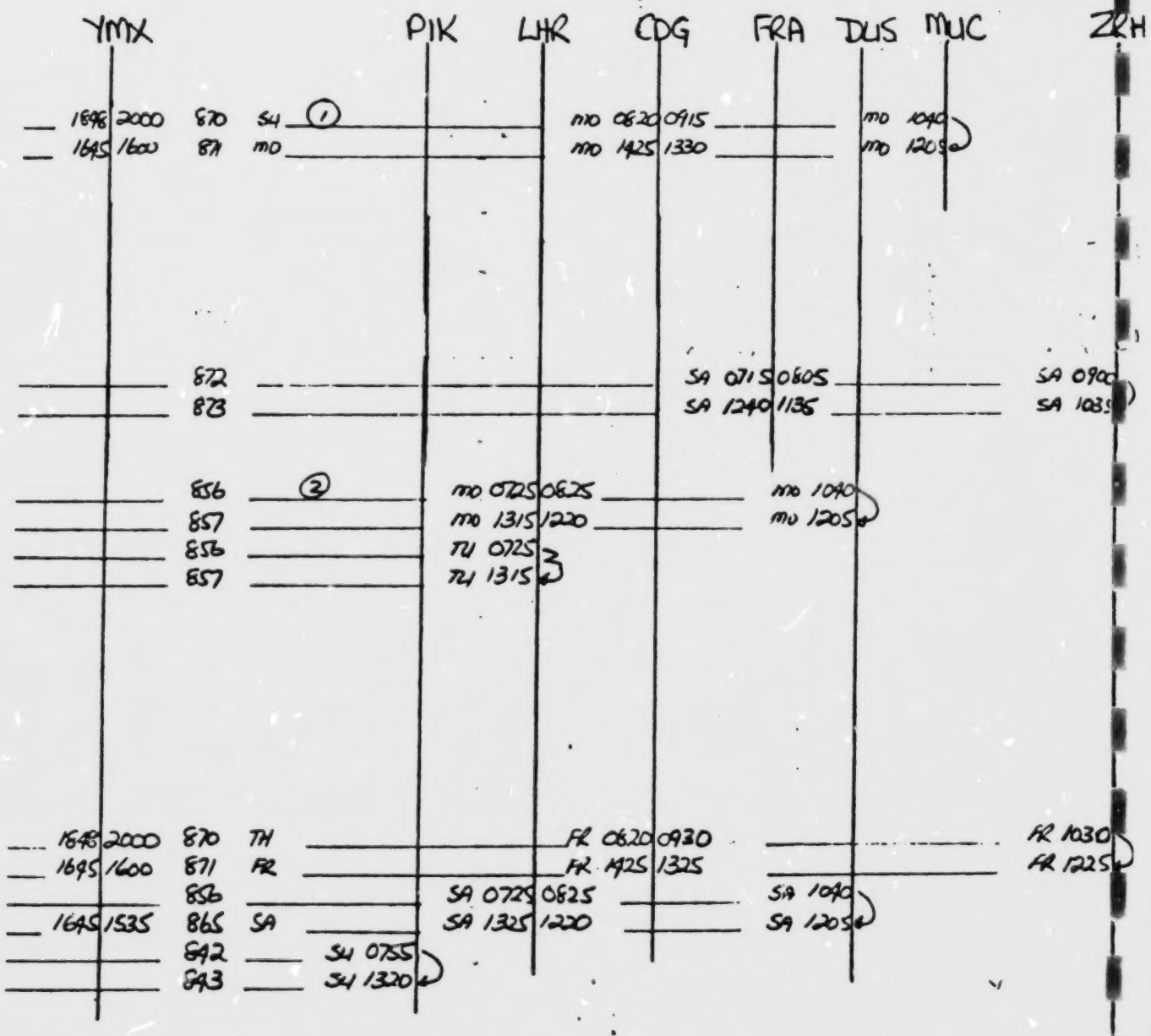


FIGURE 2-1: Reproduction d'un graphe d'allocation. Les traits verticaux représentent les aéroports et les lignes horizontales donnent la suite des vols à effectuer par un appareil. On peut voir, par exemple, dans le haut du diagramme, qu'un appareil a été affecté au vol 870 quittant Mirabel (YMX) tous les dimanches à 20:00h vers Paris (CDG) où il fait escale le lundi à 08:20h pour repartir à 09:15h vers Munich (MUC) où il arrive à 10:40h. Puis, le même appareil doit effectuer le vol 871 partant de Munich à 12:05h tous les lundis. Et ainsi de suite.

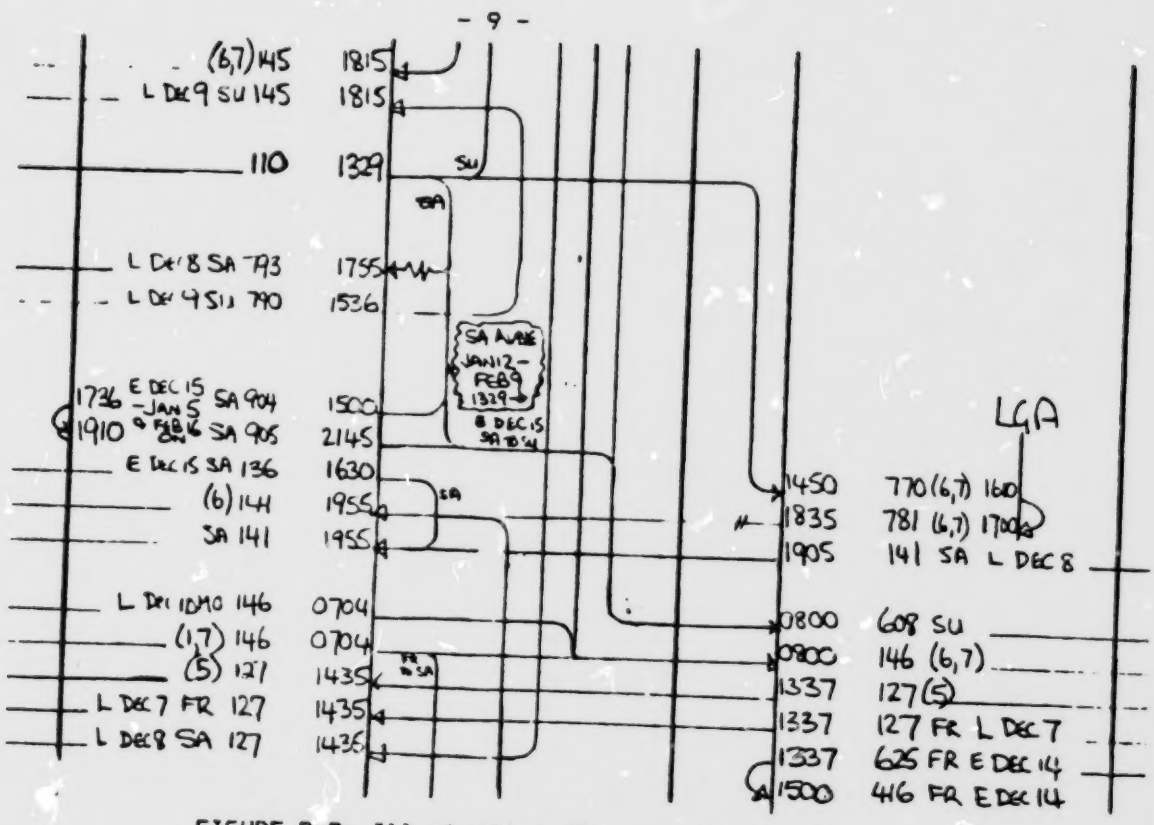


FIGURE 2-2: Illustration d'un cas d'exception.

- Le processus implique la production de plusieurs versions des graphiques. Ceux-ci étant complètement réalisés à la main, beaucoup de temps et de travail sont nécessaires à cette opération, même lorsqu'il ne s'agit que de corrections mineures d'une version à l'autre.

- La méthode est valable dans la mesure où il y a une grande stabilité dans la situation année après année. A cause de la déréglementation partielle de l'industrie aérienne au Canada, la direction d'Air Canada appréhende des modifications majeures au niveau de l'horaire. Le problème de l'allocation étant devenu de

plus en plus complexe au fil des ans, il leur serait impossible, avec le processus actuel, d'établir une toute nouvelle allocation à partir de zéro.

## 2.2 Solution envisagée

Etant donné la complexité du problème et le grand nombre d'intervenants impliqués dans le processus décisionnel (parfois avec des objectifs divergents), nous ne croyons pas qu'un modèle global d'optimisation serait adéquat. Il ne serait pas possible d'intégrer, dans un tel modèle mathématique, toutes les contraintes du problème, ou d'énoncer une fonction économique valable. De plus, un élément de solution important, trop souvent négligé par le passé lors de l'informatisation de tâches manuelles, ne pourrait pas être exploité à sa juste valeur: l'expérience des gens en place. Enfin, face à un modèle d'optimisation rigide, l'usager réagit souvent mal. A la base de toute une gamme de réactions, on peut fréquemment dégager un (ou plusieurs) des trois schèmes suivants:

-Scepticisme: Comprenant mal les méthodes et les concepts utilisés pour résoudre un problème qu'il connait très bien, l'usager ne fait pas confiance aux résultats obtenus et finit par revenir "aux bonnes vieilles méthodes".

-Crédulité: "Si ça vient de l'ordinateur, ce doit être bon." La solution trouvée par le modèle est appliquée presqu'aveuglément, sans tenir compte de tous les facteurs qui ne sont pas intégrés à ce dernier. (Les résultats ainsi obtenus contribuent forte-

ment, d'ailleurs, à augmenter le nombre d'adhérents à la tendance précédente.)

-Opportunisme: Cette approche consiste à "ajuster" les paramètres du modèle jusqu'à obtenir des solutions "sensées", c'est-à-dire celles que l'usager influent préconise depuis le début. On se retrouve ainsi avec un gros système d'optimisation utilisé de façon biaisée pour justifier des solutions (politiques) pas nécessairement optimales.

Le problème vient de ce que, trop longtemps, on a perçu l'ordinateur comme un super cerveau devant apporter l'unique solution miracle, et non comme ce qu'il doit être vraiment, un outil d'analyse permettant de choisir parmi les multiples options possibles. De plus en plus, on se rend compte de l'importance de l'intervention humaine dans les processus de planification complexes qui exigent la prise en considération de facteurs qualitatifs ou subjectifs. Aussi préconisons-nous l'implantation d'un ensemble d'outils informatiques intégrés, non pas pour remplacer et automatiser le processus de décision itératif actuel, mais dans le but d'en rendre plus efficaces et d'en accélérer les différentes étapes. Dans leur bilan des travaux réalisés dans le domaine de la planification aérienne, Etschmaier et Mathaisel (1984) en arrivent aux mêmes conclusions. Ils précisent en outre que les principales contributions apportées concernent, d'une part, dans le domaine de la recherche opérationnelle, de meilleures définitions et formalisations du processus de planification et d'autre part, dans le domaine de l'informatique, de grands progrès au niveau de l'interaction homme-machine, mais que très peu de travaux ont été faits pour une intégration de ces deux aspects du problème. Notre travail s'inscrit

justement dans ce cadre.

Deux points principaux se sont dégagés des premières discussions que nous avons eues avec le personnel d'Air Canada: d'une part le besoin, pour les différents groupes de travail impliqués dans le processus de décision, de détenir une information plus structurée et accessible, et d'autre part, pour l'équipe de planification, la nécessité de posséder un outil permettant de modifier rapidement et efficacement cette information tout en tenant compte des contraintes imposées.

Trois éléments permettront, à notre avis, d'atteindre ces objectifs:

i) Une base de données efficace, exploitant au maximum la structure particulière des données impliquées. Les horaires des vols et des allocations des appareils ont en effet une particularité importante: ils sont fortement périodiques. Un vol effectué tous les lundis, par exemple, présente une répétition dans le temps qu'il serait avantageux d'exploiter. Cependant, de nombreux cas d'exception viennent altérer quelque peu cette nature périodique. Une base de données appropriée devra donc tirer au maximum avantage de la nature périodique des données, tout en tenant compte des cas d'exception.

ii) Un ensemble de programmes interactifs-graphiques permettant de consulter et de modifier la banque de données. Nous croyons que le contexte se prête particulièrement bien à l'utilisation de techniques interactives-graphiques. D'abord, les groupes de travail de la société aérienne opèrent déjà à partir de graphiques. Ils reconnaissent déjà

l'immense avantage que présente le graphique par rapport aux tableaux numériques conventionnels pour représenter un système composé d'un grand nombre d'interactions entre des éléments de divers ensembles. Une approche graphique contribuera donc grandement à gagner la confiance des planificateurs en ne remettant pas en question de manière radicale leur méthode de travail. Seule une adaptation à des formes graphiques un peu différentes sera nécessaire.

En second lieu, cette approche permet à un usager de transmettre directement ses instructions à l'ordinateur, réduisant au minimum la nécessité de recourir à une procédure rigide. Un planificateur modifiera ainsi la banque de données en "traçant" directement sur le graphique affiché à l'écran les changements qu'il désire apporter. Il ne s'agit, somme toute, que de remplacer la table à dessin et le crayon par un écran et un dispositif d'entrée graphique (stylet lumineux, souris, curseur graphique, etc.), sans imposer au planificateur l'apprentissage de techniques informatiques compliquées. Contrairement à la table à dessin et au crayon, toutefois, ce système permettra d'effectuer des modifications instantanément et de procéder automatiquement à des vérifications pour détecter et signaler des inconsistances ou des violations de contraintes pouvant être générées par ces changements.

iii) Un ensemble d'outils analytiques pour aider le planificateur à définir une allocation minimisant le nombre d'avions requis, tout en tenant compte des contraintes liées à l'entretien. Conçus de manière à pouvoir être appliqués sur l'ensemble du problème ou sur n'importe quel sous-problème, selon les besoins du planificateur, ces outils permettront d'analyser en profondeur une situation donnée, tant au niveau

local que global. Ils pourront être utilisés en première étape pour obtenir une solution initiale, sans dépendre de l'allocation de l'année précédente et au cours du processus de production, pour évaluer différents scénarios lorsqu'une modification locale doit être effectuée.

Pour la réalisation de ce projet, nous bénéficions de l'expérience acquise par l'équipe de conception de EMME/2. EMME/2 est un logiciel interactif-graphique pour la planification du transport urbain développé au Centre de Recherches sur les Transports de l'Université de Montréal (Babin et al., 1982; Spiess, 1984). Un grand nombre de concepts développés pour EMME/2 peuvent être directement appliqués dans le cadre du présent projet. Nous disposons ainsi d'une importante bibliothèque de routines utilitaires, ce qui accélère grandement le travail de programmation. Ces routines concernent particulièrement les aspects suivants:

-Organisation logique de la banque de données

-Interface avec les divers types d'équipement

-Protocole d'interaction avec l'usager

-Graphisme

Nous traiterons de ces points de façon plus détaillée au cours des chapitres à venir.

Il a été prévu de développer et d'implanter le logiciel PLANAIR suivant un plan en quatre étapes:

i) Modélisation du problème.

L'élaboration d'un modèle permettra de cerner le problème considéré et

d'en dégager les principales composantes. Une fois ces fondements théoriques établis, il sera possible de concevoir les structures et la base de données qui constitueront un noyau autour duquel viendront se greffer les outils analytiques et interactifs-graphiques. Ce n'est que lorsque cette phase sera réalisée, alors que les "règles du jeu" auront été bien établies, que le travail de développement pourra débuter.

ii) Implantation d'un système d'accès à l'information.

Cette étape vise à mettre en place une banque de données provenant d'une situation réelle et tous les programmes donnant accès à l'information qui y est contenue, que ce soit sous forme numérique ou graphique. Il s'agit donc d'établir les bases théoriques et informatiques du problème, puis d'en évaluer la justesse et l'efficacité par l'implantation du système d'accès à l'information. A ce moment, donc, un premier système sera disponible, permettant de créer et de consulter de manière interactive une banque de données, sans pouvoir toutefois la modifier de façon efficace. Ceci implique d'abord la conception et la programmation d'un ensemble de routines utilitaires pour traiter efficacement les structures de données élaborées au cours de la première étape.

iii) Ajout des routines de modifications interactives.

Des routines interactives pour agir sur le contenu de la banque de données, numériquement (de façon conventionnelle) et graphiquement, viendront compléter le système. Nous aurons alors un produit utilisable qui pourra, dans le cadre d'une étude pilote, être employé par une société aérienne pour remplacer le processus manuel d'allocation

actuellement utilisé.

iv) Conception des outils analytiques.

Lorsque le système sera fonctionnel et utilisé en situation reelle, un ensemble d'outils analytiques adaptés aux besoins des planificateurs sera ajouté. Ces outils comprendront, entre autres, des modèles heuristiques d'allocation de même que des modèles d'optimisation des allocations. Il est important de noter la place qu'occupent ces modèles dans le système; ce sont des raffinements, des outils qui viennent s'ajouter aux autres, et qui ne constituent pas la partie prépondérante. Ainsi, l'usager les perçoit et les utilise comme ce qu'ils sont vraiment, des outils sélectifs intelligents, et non comme une boule de cristal.

Dans le cadre du présent travail, les deux premières étapes du projet ont été complétees et les deux étapes subséquentes réalisées en partie. A ce stade-ci, le logiciel PLANAIR comporte donc, en plus des programmes de création et d'accès à la banque de donnees, quelques programmes de modifications interactives et d'allocation automatique des appareils.

Ce projet a été mis en marche à la demande d'Air Canada et a progressé avec son appui. Nous avons tenu cependant à ce que le logiciel résultant puisse servir à d'autres sociétés d'aviation. Dans cette optique, il a éte jugé avantageux d'implanter, dès le début, le logiciel sur un micro-ordinateur de haut de gamme (super-micro) plutôt que sur un ordinateur de grande taille ("main-frame"). Il sera effectivement plus facile, après coup, de transporter le logiciel vers une machine de grande taille plutôt que le chemin inverse. De plus, les récents développements réalisés dans

le domaine des micro-ordinateurs permettent maintenant l'utilisation d'une très grande puissance de calcul et ce, à un coût grandement réduit.

# 3. ANALYSE ET MODELISATION DU PROBLEME

Au cours de ce chapitre, nous décrirons le modèle développé pour le problème de l'allocation des appareils d'une société aérienne à un horaire de vols à desservir. Le but premier de ce modèle est une application dans le domaine aérien. Il a toutefois été conçu pour pouvoir être appliqué au problème plus général de l'allocation des véhicules d'une flotte à une série donnée de tâches répétitives prévues entre un ensemble de points fixes prédéterminés (ports, entrepôts, etc.). Ce modèle permet de développer des algorithmes efficaces tirant avantage de la structure sous-jacente des horaires de tâches.

## 3.1 Quasi-périodicité des données

Un vol comprend un ensemble de paramètres fixes (point et heure de départ, point et heures d'arrivée). Il se répète dans le temps suivant un horaire assez périodique, mais qui peut comporter de nombreux cas d'exception. Suivant que l'on veuille faire une planification à court ou à long terme de l'horaire des vols, on sera plus ou moins intéressé à ces cas d'exception. A long terme, en effet, c'est la périodicité sous-jacente qui est plus importante tandis qu'à court terme, il est essentiel d'avoir une image très précise de l'horaire envisagé. Comme nous nous intéressons à une planification à court et moyen termes, la structure de données utilisée pour représenter les vols doit pouvoir, à la fois, exploiter la périodicité des données et tenir compte des cas d'exception.

Une première approche consiste à considérer une *semaine type* qui est ensuite répétée sur l'ensemble de la période effective du vol afin d'englober les éléments périodiques, et de garder, en plus, une liste des cas d'exception dérogeant à cette *semaine type.* C'est cette méthode qui se rapproche le plus de celle utilisée par Air Canada. L'aspect cyclique des données est ainsi exploité de manière très efficace, pour une planification à longue échéance. Cependant, à plus courte échéance, à mesure que l'on ajoute des exceptions, cette structure s'alourdit et se prête mal à un traitement systématique des données. Si l'on considère, par exemple, l'opération visant à déterminer si deux vols différents ont lieu les mêmes jours, on se rend vite compte de la grande quantité de travail nécessaire pour traiter les cas d'exception contenus dans les listes de chacun. C'est justement là une des lacunes exprimée par les gens d'Air Canada à l'égard de la méthode qu'ils emploient actuellement: les cas d'exception sont difficiles à traiter et s'intègrent mal au reste de la structure.

Une seconde méthode utilisée (Useros, 1975) consiste à subdiviser l'horaire d'un vol en une série de semaines types consécutives. On tente ainsi d'exploiter le fait que les cas d'exception peuvent parfois présenter une certaine périodicité eux aussi. Cette méthode tend à utiliser moins d'espace mémoire que la précédente et règle les cas d'exception en tentant de les incorporer dans des cycles qui leur sont propres. Toutefois, il est encore assez complexe de mettre deux vols en relation. De plus, l'ajout ou le retrait d'un élément à l'horaire d'un vol peut entraîner de grands changements dans la représentation de celui-ci, ce qui demande en retour beaucoup de travail de mise à jour pour des opérations qui sont fréquemment effectuées.

L'utilisation d'une semaine type est adéquate pour une planification à long terme, mais elle nécessite le recours à une énumération (sous une forme ou une autre) de cas d'exception pour reconstituer l'information exacte nécessaire à une planification à court terme. C'est ce travail de reconstitution qui est inefficace et empêche un traitement systématique des données. Nous avons donc choisi d'aborder le problème en sens inverse: plutôt que de garder l'information sous une forme agrégée (semaine type) et d'en dériver la forme complète (horaire exact), nous gardons la forme complète et en extrayons la forme agrégée lorsque cela est nécessaire. Nous avons ainsi opté pour une représentation explicite des vols, c'est-à-dire une représentation où chaque jour auquel doit avoir lieu un vol est conservé en mémoire. Du point de vue informatique, ceci n'est évidemment pas réalisé en utilisant un mot machine pour chaque jour d'un vol, ce qui nécessiterait beaucoup trop de mémoire et dissimulerait complètement la périodicité sous-jacente de l'horaire, mais à l'aide de vecteurs de bits. Un bit est utilisé pour chaque jour de la période couverte par l'horaire, et tous les bits des jours où un vol doit débuter sont allumés. Tous les jours de vol représentés par les bits allumés du vecteur partagent le même ensemble de paramètres fixes (point et heure de départ, point et heure d'arrivée). Ainsi, les horaires des vols sont représentés de façon très compacte à l'aide d'une structure qui reste de la même forme et de la même taille, quelle qu'en soit la composition. Cette importante propriété permet un traitement rapide et systématique de tous les vols. Utilisant ainsi des vecteurs de bits, mettre en relation deux vols devient une opération très simple et très efficace; il ne s'agit que d'effectuer des opérations logiques de base au niveau des mots machine. Non seulement n'y a-t-il aucun cas d'exception à considérer, mais en plus, les opérations logiques

sont effectuées simultanément sur tous les bits des mots machine, introduisant un parallélisme de calcul au niveau du traitement des horaires.

A l'aide d'heuristiques simples, on peut extraire de cette structure de bits la périodicité des évènements pour reformuler l'horaire en terme de semaines types. Le problème se ramène, en effet, à un problème de reconnaissance de formes dans un espace à une seule dimension, et peut être résolu assez aisément. Nous traiterons le sujet en détail dans le cadre du chapitre suivant.

## 3.2 Définition des termes et notation

Les méthodes et concepts développés ici pouvant s'appliquer à différents domaines (transport en commun, maritime, par camion, etc), nous avons établi un vocabulaire faisant abstraction du contexte aérien pour nommer les structures manipulées. Nous en introduisons ici les termes, de même qu'une notation ensembliste, que nous utiliserons par la suite:

NOEUD: (en anglais "node") Point fixe dans l'espace. Ce terme généralise la notion d'aéroport.

VEHICULE: (en anglais "vehicle") Terme généralisant la notion d'avion. Les véhicules sont partitionnés en **types** selon leurs caractéristiques techniques. A l'intérieur des types, on subdivise de nouveau les véhicules en **classes**, selon leur configuration interne. Cette configuration est variable; un véhicule peut subir des transformations et passer d'une classe à une autre. Par exemple, un DC10 passagers peut devenir un DC10 cargo suite à une reconfiguration de son espace intérieur. Les véhicules

d'une classe ne peuvent évidemment pas être reconfigurés ainsi vers n'importe quelle autre classe; un véhicule est lié à un type et ne peut passer que d'une classe de ce type à une autre classe du même type.

EVENEMENT: (en anglais "event") Tâche indivisible à laquelle doit être affecté un véhicule. L'événement est caractérisé par une position précise dans le temps (date et heure de départ, heure de fin) et dans l'espace (noeud de début, noeud de fin) et par la classe de véhicules qui l'effectue. Il faut bien prendre note du fait que ce qui se déroule entre les noeuds de début et de fin d'un événement n'est assujetti à aucune condition particulière. Ce que représente exactement l'événement est fonction du niveau d'agrégation avec lequel on veut travailler. Par exemple, on peut définir un événement comme étant un segment de vol, un vol dont on désire ignorer les escales, ou même une chaîne agrégée de vols. Un événement peut aussi représenter une période d'entretien, en lui attribuant des noeuds de début et de fin identiques.

CONNEXION: (en anglais "connection") Lien établi entre un événement se terminant à un noeud et un autre débutant au même noeud plus tard dans le temps. Une connexion représente, par exemple, le fait qu'à une date donnée, un avion ayant effectué le vol $e_1$ entre i et j doive ensuite effectuer le vol $e_2$ entre j et k.

En regroupant les événements qui ont les mêmes paramètres, on pourra n'utiliser qu'un seul ensemble de paramètres qui s'appliqueront à tout le

groupe. On développe ainsi le concept suivant:

ACTIVITE: (en anglais "activity") Ensemble d'un ou plusieurs événements effectués par la même classe de véhicule et ayant tous lieu aux mêmes heures et entre les mêmes noeuds, mais à des jours différents. On reconnaît ici une similitude avec la notion d'horaire de vol, telle que définie par les sociétés aériennes. L'activité est toutefois une notion plus générale puisqu'elle englobe aussi l'entretien et la reconfiguration des véhicules. Les activités d'entretien représentent les travaux périodiques majeurs que doivent subir préventivement les véhicules, tandis que les activités de reconfiguration représentent les périodes de temps nécessaires pour transformer un véhicule d'une classe à une autre.

De façon analogue, on regroupe des connexions sous l'appellation *accrochage:*

ACCROCHAGE: (en anglais "turn") Ensemble d'une ou plusieurs connexions, ayant lieu au même noeud, entre un sous-ensemble d'événements d'une activité $\alpha$ et un sous-ensemble d'événements d'une activité $\beta$.

En fait, on pourrait dire que les activités et les accrochages représentent des horaires alors que les événements et les connexions en sont les réalisations ponctuelles respectives.

Utilisant ces définitions, voici les notations qui seront utilisées par la suite:

$i \in N$ — noeud i appartenant à l'ensemble des noeuds N

$\alpha \in A$ — activité $\alpha$ appartenant à l'ensemble des activités A

$\alpha \in A_i^+$ , $i \in N$ — activité ayant le noeud i comme noeud de départ

$\alpha \in A_i^-$ , $i \in N$ — activité ayant le noeud i comme noeud d'arrivée

$p \in P$ — jour p de la période P couverte par l'horaire considéré. P est un ensemble ordonné de jours dans la période; des opérations mathématiques sont donc possibles sur les éléments de P.

$d_\alpha$ , $a_\alpha$ — temps de départ et d'arrivée des événements de l'activité $\alpha$, exprimés en heures et minutes

$v_\alpha^+$ , $v_\alpha^-$ — classe de véhicules effectuant l'activité $\alpha$ à son point de départ et à son point d'arrivée. (Pour la plupart des activités, on aura $v_\alpha^+ = v_\alpha^-$. La seule exception concerne les activités de reconfiguration. Dans ce dernier cas, comme il a été dit plus tôt, $v_\alpha^+$ et $v_\alpha^-$ doivent appartenir au même type de véhicule.)

$\lambda_\alpha$ — nombre de changements de date séparant le début et la fin d'un événement de l'activité $\alpha$. Si $\alpha$ débute et se termine le même jour, $\lambda_\alpha = 0$. Si $\alpha$ se termine le lendemain, $\lambda_\alpha = 1$, etc..

$\delta_\alpha^p$ — Variable 0-1 indiquant si un événement de l'activité $\alpha$ débute au jour p de la période ou non. Par extension, nous

dénoterons par $\delta_\alpha$ le vecteur des variables $\delta_\alpha^p$ , $p \in P$.

$e=(\alpha,p)$ événement de l'activité $\alpha$ débutant au jour p de la période. Evidemment, l'existence de $e=(\alpha,p)$ implique $\delta_\alpha^p=1$ et vice versa.

$\tau \in T$ accrochage $\tau$ appartenant à l'ensemble des accrochages T

$\tau=(\alpha,\beta,\lambda_\tau)$ accrochage entre des événements de l'activité $\alpha \in A_i^-$ et des événements de l'activité $\beta \in A_i^+$ survenant $\lambda_\tau$ jours plus tard

$\delta_\tau^p$ variable 0-1 indiquant si une connexion de l'accrochage $\tau$ débute au jour p de la période ou non. Par extension, nous dénoterons par $\delta_\tau$ le vecteur des variables $\delta_\tau^p$ , $p \in P$.

$c=(\tau,p)$ connexion de l'accrochage $\tau$ débutant au jour p de la période. Evidemment, l'existence de $c=(\tau,p)$ implique $\delta_\tau^p=1$ et vice versa.

$\tau \in T_\alpha^-$ accrochage $\tau$ appartenant à l'ensemble $T_\alpha^-$ des accrochages prédécesseurs de l'activité $\alpha \in A_i^+$, c'est-à-dire les accrochages de la forme $\tau=(\beta,\alpha,\lambda_\tau)$ , $\beta \in A_i^-$

$\tau \in T_\alpha^+$ accrochage $\tau$ appartenant à l'ensemble $T_\alpha^+$ des accrochages successeurs de l'activité $\alpha \in A_i^-$, c'est-à-dire les accrochages de la forme $\tau=(\alpha,\beta,\lambda_\tau)$ , $\beta \in A_i^+$

Nous avons défini, jusqu'ici, les variables booléennes $\delta_\alpha^p$ et $\delta_\tau^p$. Par la suite, nous emploierons quelquefois la notation $\delta_x^p$ pour désigner

indifféremment l'une ou l'autre de ces variables. De même pour les vecteurs $\delta_\alpha$ et $\delta_\tau$ avec $\delta_x$.

### 3.3 Opérateurs et propriétés de base

Pour manipuler les entités que nous venons d'énoncer, nous définirons ici un ensemble d'opérateurs mathématiques, puis dériverons à l'aide de ces derniers une série de propriétés de base qui formeront les fondements analytiques de notre modèle.

Trois types d'opérateurs sont nécessaires: des opérateurs logiques, de décalage et de cardinalité. En voici les définitions:

- Les variables $\delta_x^p$ étant des variables booléennes (0-1), on peut utiliser des opérateurs logiques pour les manipuler. On dénote ainsi par $\wedge$, $\vee$ et $\bullet$ les opérateurs logiques **et, ou** et ou **exclusif** appliqués aux variables $\delta_x^p$ et aux vecteurs $\delta_x$. De plus, nous dénoterons par $\cap$ et $\cup$ la généralisation des opérateurs **et** et **ou** à des ensembles de variables. Par exemple, soit T, l'ensemble des accrochages $\{\tau_1, \tau_2, \ldots, \tau_n\}$. Alors,

$$\delta_{\tau_1} \vee \delta_{\tau_2} \vee \ldots \vee \delta_{\tau_n} = \bigcup_{\tau \in T} \delta_\tau$$

De même pour $\cap$ avec l'opérateur **et**.

- Il sera fréquemment nécessaire, à partir d'un vecteur $\delta_x$, d'obtenir un vecteur $\bar{\delta}_x$ décalé de plus ou moins $\lambda$ jours par rapport à $\delta_x$. Il nous faut donc définir un opérateur de décalage qui ajoutera ou retranchera $\lambda * 24$ heures aux heures de départ et d'arrivée de tous les $\delta_x^p$. Empruntant une notation utilisée dans le langage C, nous utiliserons le signe >> pour dénoter un décalage positif des éléments du vecteur $\delta_x$ (i.e.

décalage vers le futur) et << pour dénoter un décalage négatif. Par exemple, $\delta_\alpha$ représente l'ensemble des dates de début des événements de $\alpha$. On peut obtenir, en décalant celui-ci de $\lambda_\alpha$ jours, l'ensemble des dates de fin. $\bar{\delta}_\alpha = \delta_\alpha >> \lambda_\alpha$ représente donc cet ensemble des dates de fin. De cette façon,

$$\bar{\delta}_\alpha = \delta_\alpha >> \lambda_\alpha \quad \text{est équivalent à} \quad \delta_\alpha^p = \bar{\delta}_\alpha^{p+\lambda_\alpha} \qquad \forall\, p \varepsilon P$$

- Enfin, $||\delta_x||$ indiquera la cardinalité du vecteur $\delta_x$, c'est-à-dire le nombre de $\delta_x^p$ qui ont une valeur non nulle.

Par opposition à une chronologie réelle basée sur les événements, nous emploierons le terme *chronologie horaire* pour désigner un ordonnancement basé sur les activités; c'est-à-dire que le facteur d'ordonnancement est l'heure de départ ou d'arrivée d'une activité, sans tenir compte des dates où cela doit avoir lieu. Cette chronologie se répète cycliquement, en ce sens qu'une activité dont l'heure de départ est $d_\alpha$ pourrait aussi survenir 24 heures plus tard. Ainsi, chaque période de 24 heures est suivie d'une autre identique. Nous utiliserons $\alpha(\lambda)$ pour désigner, selon cette chronologie horaire, une activité $\alpha$ qui surviendrait avec $\lambda$ jours de différence par rapport à une autre, c'est-à-dire une activité à laquelle on aura ajouté $\lambda * 24$ heures à $d_\alpha$ et $a_\alpha$. Il va de soi que les notations $\alpha$ et $\alpha(0)$ sont équivalentes. Illustrons cette notion à l'aide d'un exemple: soient $\alpha \varepsilon A_i^-$ avec $a_\alpha$ = 22h. et $\beta \varepsilon A_i^+$ avec $d_\beta$ = 10h.. Il est certain, si elles surviennent les mêmes journées, que ces deux activités ne peuvent pas être accrochées puisque $\alpha$ arrive après que $\beta$ soit partie. Par contre, si l'on considère que le départ de $\beta$ peut avoir lieu à dix heures, mais le lendemain ($\beta(1)$), il existe alors une possibilité de pouvoir créer un accro-

chage entre ces deux activités. Par conséquent, il ne sert à rien d'essayer de créer l'accrochage $\tau=(\alpha,\beta,\lambda_\tau)$ avec $\lambda_\tau=0$, mais on peut tenter de créer ce même accrochage avec $\lambda_\tau>0$.

Nous avons maintenant en main les outils nous permettant de définir des relations fondamentales et d'en déduire un ensemble de propriétés de base. Ces dernières nous seront utiles, par la suite, pour nous assurer de la cohérence des données et concevoir les algorithmes d'allocation.

Nous dirons qu'un événement $e=(\beta,p)$ est *pré-accroché* s'il existe, au noeud de départ de celui-ci, une connexion $c=(\tau,p)$ , $\tau=(\alpha,\beta,\lambda_\tau)$, le reliant à un autre événement. C'est-à-dire,

$$e=(\beta,p) \text{ est pré-accroché} \quad \langle == \rangle \quad \exists \quad \tau \in T_\beta^- \text{ tel que } \delta_\tau^p = 1$$

Symétriquement, nous dirons qu'un événement $e=(\alpha,p)$ est *post-accroché* s'il existe, au noeud d'arrivée de celui-ci, une connexion $c=(\tau,p)$ , $\tau=(\alpha,\beta,\lambda_\tau)$, le reliant à un autre événement. C'est-à-dire,

$$e=(\alpha,p) \text{ est post-accroché} \quad \langle == \rangle \quad \exists \quad \tau \in T_\alpha^+ \text{ tel que } \delta_\tau^p = 1$$

L'activité $\alpha$ sera *pré-saturée* lorsque tous les événements qui la composent seront pré-accrochés. C'est-à-dire, si l'égalité suivante est satisfaite:

$$\delta_\alpha = \bigcup_{\tau \in T_\alpha^-} (\delta_\tau >> \lambda_\tau)$$

On définit symétriquement le terme *post-saturé:*

$$\delta_\alpha >> \lambda_\alpha = \bigcup_{\tau \in T_\alpha^+} \delta_\tau$$

Enfin, l'activité $\alpha$ est dite *saturée* si elle est à la fois pré- et post- saturée.

Une activité $\beta(\lambda)$ est dite *compatible* avec l'activité $\alpha$ si elle satisfait les conditions suivantes:

i) $\alpha$ et $\beta(\lambda)$ ont lieu au même noeud i et

$$\beta \varepsilon A_i^+ \quad \text{si} \quad \alpha \varepsilon A_i^-$$
$$\beta \varepsilon A_i^- \quad \text{si} \quad \alpha \varepsilon A_i^+$$

ii) La classe de véhicules effectuant $\alpha$ est la même que celle effectuant $\beta$:

$$v_\alpha^- = v_\beta^+ \quad \text{si} \quad \alpha \varepsilon A_i^-$$
$$v_\alpha^+ = v_\beta^- \quad \text{si} \quad \alpha \varepsilon A_i^+$$

iii) Le départ de l'activité qui débute au noeud i survient après l'arrivée de celle qui s'y termine:

$$\lambda \geq 0 \quad \text{et} \quad d_{\beta(\lambda)} = d_\beta + \lambda \$24 > a_\alpha \qquad \text{si } \alpha \varepsilon A_i^-$$
$$\lambda \leq 0 \quad \text{et} \quad d_\alpha > a_{\beta(\lambda)} = a_\beta + \lambda \$24 \qquad \text{si } \alpha \varepsilon A_i^+$$

iv) Les dates de fin de l'activité qui se termine ont une intersection non nulle avec les dates de début de celle qui commence, considérant la période de $\lambda$ jours qui les sépare:

$$\{ \delta_\alpha >> \lambda_\alpha \} \wedge \{ \delta_\beta << \lambda \} \neq \emptyset \qquad \text{si } \alpha \varepsilon A_i^-$$
$$\{ \delta_\beta >> \lambda_\beta \} \wedge \{ \delta_\alpha << -\lambda \} \neq \emptyset \qquad \text{si } \alpha \varepsilon A_i^+$$

Nous dénoterons par $\beta \varepsilon A_i^-(\alpha)$ le sous-ensemble des activités $\beta(\lambda) \varepsilon A_i^-$

compatibles avec $\alpha \varepsilon A_i^+$. Symétriquement, $\beta \varepsilon A_i^+(\alpha)$ dénotera le sous-ensemble des activités $\beta(\lambda) \varepsilon A_i^+$ compatibles avec $\alpha \varepsilon A_i^-$.

Un accrochage $\tau=(\alpha,\beta,\lambda_\tau)$ établit une correspondance entre les événements de $\alpha$ et ceux de $\beta$. Pour qu'un tel accrochage soit valide, quatre conditions doivent être satisfaites:

i) $\alpha$ et $\beta(\lambda_\tau)$ sont compatibles, c'est-à-dire:

$$\beta(\lambda_\tau) \varepsilon A_i^+(\alpha)$$

ii) Soient $\bar{\delta}_\alpha = \delta_\alpha >> \lambda_\alpha$, les dates de fin de $\alpha$ et $\bar{\delta}_\beta = \delta_\beta << \lambda_\tau$, $\delta_\tau$ doit être un sous-ensemble non vide de l'intersection entre $\bar{\delta}_\alpha$ et $\bar{\delta}_\beta$:

$$\{\delta_\tau \wedge (\bar{\delta}_\alpha \wedge \bar{\delta}_\beta)\} = \delta_\tau \quad , \delta_\tau \neq \emptyset \tag{1}$$

iii) Chaque événement de $\alpha$ ne peut être post-accroché qu'à au plus un autre événement. L'inverse impliquerait qu'un véhicule doive, après avoir effectué un événement de $\alpha$, se subdiviser pour effectuer simultanément plusieurs tâches. Ainsi, $\tau$ ne doit pas entrer en conflit avec un autre accrochage successeur de $\alpha$:

$$\delta_\tau \wedge \delta_x = 0 \qquad \forall\, x \varepsilon T_\alpha^+ \tag{2}$$

iv) De la même façon, chaque événement de $\beta$ ne peut être pré-accroché qu'à au plus un autre événement. L'inverse impliquerait que plusieurs véhicules doivent effectuer simultanément la même tâche. Ainsi, $\tau$ ne doit pas, non plus, entrer en conflit avec un autre accrochage prédécesseur de $\beta$:

$$\{\delta_\tau >> \lambda_\tau\} \wedge \{\delta_x >> \lambda_x\} = 0 \qquad \forall\, x \varepsilon T_\beta^- \tag{3}$$

Il est important de remarquer que ces quatre conditions de validité interviennent au niveau des activités et des accrochages, sans avoir à *descendre* au niveau plus désagrégé des événements et des connexions. Jusqu'ici, nous avons donc réussi, dans une certaine mesure, à faire abstraction des composantes des vecteurs $\delta_x$. On peut ainsi vérifier la validité d'un accrochage en traitant en bloc les vecteurs $\delta_x$, sans devoir traiter séparément une à une les variables $\delta_x^p$. Comme nous l'avons laissé entendre au début de ce chapitre, les vecteurs $\delta_x$ seront implantés à partir de vecteurs de bits. On entrevoit donc ici la possibilité de travailler directement au niveau des mots machine sans devoir tenir compte de la suite de bits qui les composent, entraînant un certain parallélisme de calcul.
Il faut maintenant réussir à générer toutes les connexions, en restant toujours au niveau plus agrégé des activités. C'est à ce résultat que nous tenterons d'arriver dans la prochaine section.

## 3.4 Génération des accrochages

La création des connexions se ramène au problème suivant: pour chaque départ prévu à un noeud, quels sont les véhicules (i.e. les arrivées) disponibles et lequel choisir? La plupart des algorithmes qui ont été développés par les sociétés aériennes pour obtenir une solution initiale réalisable, sont de type LIFO, FIFO ou une combinaison de ceux-ci (Barnard, 1980). Le LIFO a tendance à maximiser la variance des temps de connexions. Cette particularité permet d'intercaler plus facilement des périodes d'entretien car elles nécessitent de longues périodes de stationnement au sol. Il est aussi plus facile d'effectuer des changements opérationnels lorsque des délais à l'horaire prévu sont encourus. Conséquemment, le LIFO occupe une place importante dans l'élaboration de solutions initiales pour les noeuds où se font les entretiens des véhicules.

Nous avons donc entrepris de développer d'abord un algorithme de type LIFO qui puisse exploiter la structure des vecteurs de bits utilisés et tirer avantage de la périodicité des horaires lorsque celle-ci est présente. Nous exposerons cet algorithme dans les pages qui suivent.

Sur la base de la chronologie horaire décrite précédemment, définissons d'abord deux fonctions établissant des relations entre les activités se déroulant à un noeud:

- Soit la fonction $ARR^-(\alpha)$ qui détermine l'activité $\beta \varepsilon A_i^-$ qui précède immédiatement $\alpha$ selon la chronologie horaire. Plus précisément,

$$ARR^-(\alpha) = \left\{ \beta(\lambda) \varepsilon A_i^- \;\middle|\; \lambda \leq 0; \quad \min_{\gamma(\lambda) \varepsilon A_i^-} \left\{ d_\alpha - a_{\gamma(\lambda)} \right\} = d_\alpha - a_{\beta(\lambda)} > 0 \right\} \quad \text{si } \alpha \varepsilon A_i^+ \quad (4)$$

$$ARR^{-}(\alpha) = \{\beta(\lambda)\varepsilon A_i^{-} \mid \lambda \leq 0;\ \min_{\gamma(\lambda)\varepsilon A_i^{-}} \{a_\alpha - a_{\gamma(\lambda)}\} = a_\alpha - a_{\beta(\lambda)} > 0\} \quad \text{si } \alpha\varepsilon A_i^{-} \qquad (5)$$

La fonction retournera zéro lorsque l'ensemble solution sera vide. On pourrait définir symétriquement la fonction $ARR^{+}(\alpha)$ pour désigner l'activité $\beta\varepsilon A_i^{-}$ succédant immédiatement $\alpha$.

- Soit aussi la fonction $DEP^{+}(\alpha)$ qui détermine l'activité $\beta\varepsilon A_i^{+}$ qui succède immédiatement $\alpha$ selon la chronologie horaire. Plus précisément,

$$DEP^{+}(\alpha) = \{\beta(\lambda)\varepsilon A_i^{+} \mid \lambda \geq 0;\ \min_{\gamma(\lambda)\varepsilon A_i^{+}} \{d_{\gamma(\lambda)} - d_\alpha\} = d_{\beta(\lambda)} - d_\alpha > 0\} \quad \text{si } \alpha\varepsilon A_i^{+} \qquad (6)$$

$$DEP^{+}(\alpha) = \{\beta(\lambda)\varepsilon A_i^{+} \mid \lambda \geq 0;\ \min_{\gamma(\lambda)\varepsilon A_i^{+}} \{d_{\gamma(\lambda)} - a_\alpha\} = d_{\beta(\lambda)} - a_\alpha > 0\} \quad \text{si } \alpha\varepsilon A_i^{-} \qquad (7)$$

Ici encore, on pourrait définir symétriquement la fonction $DEP^{-}(\alpha)$ pour désigner l'activité $\beta\varepsilon A_i^{+}$ précédant immédiatement $\alpha$.

Nous restreindrons encore la portée des fonctions ARR et DEP en exigeant, en plus, que l'activité $\beta(\lambda)$ soit compatible avec $\alpha$, c'est-à-dire $\beta(\lambda)\varepsilon A_i(\alpha)$. On définit ainsi les fonctions $\overline{ARR}$ et $\overline{DEP}$:

$$\overline{ARR}^{-}(\alpha) = \{\beta(\lambda)\varepsilon A_i^{-}(\alpha) \mid \min_{\gamma(\lambda)\varepsilon A_i^{-}(\alpha)} \{d_\alpha - a_{\gamma(\lambda)}\} = d_\alpha - a_{\beta(\lambda)}\} \qquad (8)$$

Ici, on doit nécessairement avoir $\alpha\varepsilon A_i^{+}$, par la condition i) de compatibilité. On définit symétriquement $\overline{DEP}^{+}(\alpha)$.

Notre but est la réduction de la quantité de calculs nécessaire à la création des accrochages en travaillant au niveau des activités et en

traitant en bloc tous les événements composant celle-ci plutôt que séparément un à un. La proposition suivante nous permettra d'arriver à nos fins:

PROPOSITION: Soient deux activités $\alpha \varepsilon A_i^-$ et $\beta \varepsilon A_i^+$ , $i \varepsilon N$, qui ne font partie d'aucun accrochage au noeud i, c'est-à-dire,

$$T_\alpha^+ = \emptyset$$

$$T_\beta^- = \emptyset$$

Si $\alpha$ et $\beta$ satisfont les deux conditions suivantes:

$$\overline{\overline{ARR}}^-(\beta(\lambda)) = \alpha \quad ,$$

$$\overline{\overline{DEP}}^+(\alpha) = \beta(\lambda) \quad ,$$

alors l'accrochage $\tau=(\alpha,\beta,\lambda)$ défini par

$$\delta_\tau = ( \delta_\alpha >> \lambda_\alpha ) \wedge ( \delta_\beta << \lambda )$$

est valide et toutes les connexions qui le composent respectent la stratégie LIFO.

PREUVE:

1) $\tau$ est réalisable:

Par définition de $\overline{\overline{ARR}}$ et $\overline{\overline{DEP}}$ , $\alpha$ et $\beta(\lambda)$ sont compatibles, satisfaisant la condition de validité i) des accrochages.

La condition ii) est aussi satisfaite puisque

a) $\delta_\tau = ( \delta_\alpha >> \lambda_\alpha ) \wedge ( \delta_\beta << \lambda ) \neq \emptyset$ car $\alpha$ et $\beta(\lambda)$ sont compatibles

b) comme $\lambda_\tau = \lambda$, on a

$$\delta_\tau \wedge ( \bar{\delta}_\alpha \wedge \bar{\delta}_\beta ) = \delta_\tau$$

Enfin, les conditions iii) et iv) sont nécessairement satisfaites, puisqu'au départ, $T_\alpha^+ = \emptyset$ et $T_\beta^- = \emptyset$.

2) Les connexions de $\mathcal{T}$ respectent la stratégie LIFO:

Par contradiction, supposons maintenant qu'il existe une connexion $c=(\mathcal{T},p)$ qui ne satisfasse pas la stratégie LIFO. On a alors forcément une des deux situations suivantes:

i) $\exists\ \beta_1(\mu)\in A_i^-(\alpha) \mid a_\alpha < a_{\beta_1(\mu)} < d_{\beta(\lambda)}$

Ce qui entre en contradiction avec la définition même de $\overline{ARR}^-(\beta(\lambda))$ qui exige que

$$d_{\beta(\lambda)} - a_\alpha = \min_{\gamma(\mu)\in A_i^-(\beta(\lambda))} \{ d_{\beta(\lambda)} - a_{\gamma(\mu)} \}$$

ii) $\exists\ \beta_2(\mu)\in A_i^+(\alpha) \mid a_\alpha < d_{\beta_2(\mu)} < d_{\beta(\lambda)}$

Ce qui entre en contradiction avec la définition même de $\overline{DEP}^+(\alpha)$ qui exige que

$$d_{\beta(\lambda)} - a_\alpha = \min_{\gamma(\mu)\in A_i^+(\alpha)} \{ d_{\gamma(\mu)} - a_\alpha \} \quad \blacksquare$$

La stratégie LIFO est "sans mémoire", en ce sens qu'elle ne garde aucune information d'une connexion qui a été créée, si ce n'est le fait que les événements impliqués ne sont plus disponibles pour d'autres connexions. On peut donc, dès qu'une connexion a été créée, retirer les événements impliqués, sans changer le reste du problème. Pour générer les accrochages suivant la stratégie LIFO, on peut donc se servir du résultat de la proposition pour créer un premier accrochage et réduire le problème en retirant les événements impliqués, afin de pouvoir appliquer de nouveau la procédure d'accrochage qui est à la base de la proposition. De cette manière, il est possible de construire des accrochages suivant la stratégie LIFO, à partir

de conditions imposées sur les activités à relier, et non sur les événements. On construit ainsi simultanément toutes les connexions qui composent un accrochage. Dans le pire des cas, chaque accrochage peut n'être composé que d'une seule connexion. Ceci peut survenir si, par exemple, les activités à relier sont composées d'événements à peu près aléatoires, ne comportant aucune périodicité. On se retrouve alors à plus ou moins traiter les événements un à un. A l'autre extrême cependant, et c'est là que le plus grand nombre de cas se retrouvent, plus les activités auront une nature périodique, plus elles auront tendance à se "ressembler", et plus il y aura de parallélisme au niveau de la connexion des événements.

L'algorithme suivant, basé sur le résultat de la proposition que nous venons d'énoncer, construit l'ensemble des accrochages suivant la stratégie LIFO. Nous supposerons ici que les fonctions ARR et DEP retournent zéro lorsque l'activité recherchée n'existe pas. Nous définirons de plus une quantité $\bar{\lambda}$ qui représente le nombre maximal de jours permis pour la durée d'un accrochage.

**Algorithme A1:**

ETAPE 0: $\beta(\lambda) := \beta_0$ ($\beta_0$ = premier départ au noeud $i \in N$ considéré)

ETAPE 1: Tantque ( $\lambda < \bar{\lambda}$ et $\beta(\lambda) \neq 0$ )

$\alpha := ARR^{-}(\beta(\lambda))$

$\bar{\delta}_\beta := \{ \delta_\beta \bullet \{ \bigcup_{\tau \in T_\beta^-} ( \delta_\tau >> \lambda_\tau ) \} \} << \lambda$

Tantque ( $\alpha \neq 0$ et $||\bar{\delta}_\beta|| \neq 0$ ) faire

$\bar{\delta}_\alpha := (\delta_\alpha >> \lambda_\alpha) \bullet \{ \bigcup_{\tau \in T_\alpha^+} \delta_\tau \}$

Si ( $v_\alpha^- = v_\beta^+$ et $\left|\left|\bar{\delta}_\alpha \wedge \bar{\delta}_\beta\right|\right| \neq 0$ ) alors

a) Créer accrochage $\tau=(\alpha,\beta,\lambda)$:

$\delta_\tau := \bar{\delta}_\alpha \wedge \bar{\delta}_\beta$

$\lambda_\tau := \lambda$

b) Réduire problème:

$\bar{\delta}_\beta := \bar{\delta}_\beta \bullet \delta_\tau$

$\bar{\delta}_\alpha := \bar{\delta}_\alpha \bullet \delta_\tau$

Fin (Si)

$\alpha := ARR^-(\alpha)$

Fin (Tantque)

$\beta(\lambda) := DEP^+(\beta(\lambda))$

Fin (Tantque)

L'algorithme est composé de deux boucles imbriquées. La boucle interne tente de pré-saturer une activité $\beta(\lambda) \varepsilon A_i^+$ en la comparant successivement aux activités $\alpha \varepsilon A_i^-$ qui la précèdent (du plus près au plus loin) dans la chronologie horaire. La boucle externe parcourt tous les $\beta(\lambda) \varepsilon A_i^+$ selon l'ordre chronologique horaire ascendant. Cette façon de parcourir les $\alpha \varepsilon A_i^-$ et les $\beta \varepsilon A_i^+$ fait en sorte qu'au départ, les conditions plus faibles

$$ARR^-(\beta(\lambda)) = \alpha$$

$$DEP^+(\alpha) = \beta(\lambda)$$

sont satisfaites. On cherche ensuite un $\alpha \varepsilon A_i^-$ qui satisfasse les conditions plus fortes

$$\overline{ARR^-}(\beta(\lambda)) = \alpha$$

$$\overrightarrow{DEP}(\alpha) = \beta(\lambda).$$

Dans le pire des cas, avant d'être pré-saturée, un $\beta(\lambda) \varepsilon A_i^+$ sera ainsi comparé à toutes les activités $\alpha \varepsilon A_i^-$ qui le précèdent dans la chronologie horaire. Ce genre de permutation nous assure que tous les accrochages qu'il est possible de construire le seront.

Nous verrons, au chapitre suivant, comment les fonctions ARR et DEP peuvent être implantées à l'aide de simples structures de listes chaînées.

# 4. ASPECTS INFORMATIQUES

Dans ce chapitre, nous montrerons comment les concepts développés au chapitre précédent ont été appliqués et implantés. Nous traiterons, dans un premier temps, de certains détails liés à l'implantation des vecteurs de bits, puis décrirons l'organisation logique de la base de données, les structures de données utilisées de même que certains algorithmes de base.

## 4.1 Implantation des vecteurs de bits

Comme nous l'avons vu plus tôt, les vecteurs $\delta_x$ qui représentent l'horaire complet de chaque activité et de chaque accrochage sont implantés à partir de vecteurs de bits. Un bit est utilisé pour chaque jour de la période de planification et indique si l'activité ou l'accrochage a lieu ou non à cette date. De cette façon, pour une période de planification d'un an, 366 bits (soit 12 mots sur une machine à 32 bits) sont nécessaires pour chaque activité et chaque accrochage de la banque de données. Du point de vue informatique, l'implantation des opérations $\wedge$, $\vee$ et $\bullet$ pour manipuler ces vecteurs est immédiate puisque l'on peut appliquer presque directement les opérations de base utilisées par la machine de façon indépendante sur chacun des mots du vecteur. Pour l'opération de décalage (>>), toutefois, la situation est bien différente. En effet, on ne peut tout bonnement appliquer l'opération de décalage séparément sur chacun des mots du vecteur, puisque les bits décalés sortants ("shift out bits") sont perdus alors qu'il faudrait qu'ils "passent" au mot adjacent dans le vecteur.

Nous voulons appliquer le décalage à un vecteur complet, en le traitant d'un bloc, comme s'il ne s'agissait que d'un seul mot. Il a donc fallu récrire une routine de décalage plus adaptée qui simule des mots machine composés d'un nombre arbitraire de bits. Mais que faire, alors, avec les bits décalés qui dépassent la limite du vecteur? Faut-il les conserver quelque part, effectuer un décalage circulaire, ou simplement les laisser tomber? Pour répondre à cette question, il ne faut pas perdre de vue le contexte dans lequel sont utilisés les bits et ce qu'ils sont censés représenter. Chaque bit symbolise une date à l'intérieur d'une période de temps donné et nous utilisons le décalage pour avancer ou reculer l'horaire d'une activité ou d'un accrochage de $\lambda$ jours. Vu de cette manière, un bit qui est décalé en dehors du vecteur $\delta_x$ représente donc une date qui n'est pas comprise dans la période de planification considérée. Comme on sait qu'il y aura souvent de grandes différences entre les horaires de deux périodes consécutives, on ne peut prendre pour acquis qu'une activité prévue à l'horaire de l'une est aussi prévue à l'horaire de l'autre. Dès lors, nous sommes d'avis que la manière correcte de traiter les bits décalés qui sortent du vecteur est de les laisser tomber. Toute autre action équivaudrait à faire des hypothèses sur les horaires d'une période dont on ne sait, en fait, rien. Ce choix a comme conséquence directe que l'opération de décalage n'est pas nécessairement réversible:

$$\bar{\delta}_x = (\delta_x >> \lambda) >> -\mu \quad \text{n'implique pas} \quad \bar{\delta}_x = \delta_x >> (\lambda - \mu)$$

Pour évidente qu'elle soit, énoncée ainsi, cette propriété n'en demeure pas moins importante. Il faudra en effet y prendre garde afin d'éviter de perdre de l'information.

Un autre point important concerne l'encodage et le décodage des vecteurs de bits. Les horaires des activités et des accrochages sont spécifiés à l'aide d'un énoncé composé de trois éléments: une date de début, une date de fin et un ensemble de jours dans la semaine. Un tel énoncé peut être, par exemple, "JA01 JL31 1" pour indiquer qu'une activité a lieu tous les lundis de janvier à juillet. (On utilise la convention souvent employée dans les horaires où 1=lundi, 2=mardi, etc.) Plusieurs de ces énoncés peuvent être utilisés pour définir l'horaire d'une activité. Il est assez facile d'analyser l'information contenue dans ces énoncés pour la transformer et l'encoder sous forme de vecteurs de bits. Le chemin inverse est cependant beaucoup plus complexe. Comment, à partir d'une suite quelconque de bits allumés et éteints, reconstituer l'information pour la présenter dans un format plus accessible à l'usager, c'est-à-dire sous forme d'énoncés? C'est en fait un problème très intéressant de reconnaissance de formes dans un espace à une seule dimension; dans la suite linéaire de bits, il faut arriver à déceler et reconnaître les groupes de bits, ou formes, qui se répètent pour ensuite les exprimer par des énoncés. Voici, par exemple, une suite de bits qui représente l'horaire d'une activité à l'intérieur d'une période débutant un samedi premier janvier:

110000011010001100000110000011000000...000

Vu ainsi dans son ensemble, il est facile d'y reconnaître l'horaire d'une activité dont les événements ont lieu toutes les fins de semaine de janvier, avec un événement additionnel le 11 janvier. On pourrait alors traduire ceci par

```
JA01 JA30 67
AD   JA11
```

où AD indique un événement additionnel.

Une des difficultés du problème réside dans l'identification des cas d'exception qui cachent la périodicité sous-jacente. Ainsi, dans l'exemple précédent, avant d'être en mesure de dire que le 11 janvier constitue une exception, il faut examiner la suite de bits dans son ensemble, afin d'avoir une vue globale du problème. Afin d'y arriver, nous avons imaginé une méthode heuristique qui divise le problème en deux parties: un premier parcours pour avoir une vue globale, et un second pour identifier les cas d'exception. Nous donnons les détails de cette méthode en l'appliquant à l'exemple utilisé précédemment:

1) A partir d'une période de base de sept jours, on découpe la suite de bits en sections composées de périodes consécutives identiques. On obtient ainsi

$$1100000\,|\,1101000\,|\,11000001100000011000000\,|\,0\ldots0$$

Si la suite de bits ne comporte aucune périodicité, alors, au pire cas, on la divisera en sections ne comprenant chacune que 7 bits, ce qui revient à énumérer chaque semaine de l'horaire. C'est encore la meilleure approche à utiliser pour ces cas extrêmes.

2) Si elles ne présentent pas une forme trop différente, on tente d'incorporer aux sections composées du plus grand nombre de périodes les sections adjacentes plus petites. Ces dernières deviendront alors des exceptions. On aura par exemple

$$1100000\,|\,11010001100000110000011000000\,|\,0\ldots0$$

exception

Puis finalement

1100000110100011000001100000011000000|0...0
^
exception

Deux points restent à éclaircir:

- A partir de quand doit-on considérer qu'une section constitue une exception plutôt qu'une coupure dans l'horaire?

- Comment quantifier l'expression "forme pas trop différente" employée précédemment à l'étape 2?

Le but de l'opération est de présenter l'information dans la forme la plus compacte possible. On doit donc choisir les critères et les paramètres en fonction de cet objectif à atteindre. Considérons un nouvel exemple où les deux séries d'énoncés suivantes représentent le même horaire et sont équivalentes:

```
MA01 DE31 3          MA01 JN30 3
AD   JL07            JL01 JL31 35
AD   JL14            AU01 DE31 3
AD   JL21
AD   JL28
```

La deuxième série est cependant plus appropriée, car elle est plus compacte et plus facile à interpréter. De façon générale, dès que les exceptions s'étendent sur plus de deux semaines ou qu'il y en a plus de deux à l'intérieur d'une même semaine, il devient avantageux de les considérer plutôt comme des sections indépendantes.

L'algorithme suivant formalise cette méthode pour décoder un vecteur $\delta_x$ couvrant une période allant du jour 1 à "finperiode". Nous utiliserons

le tableau "forme[]" afin de stocker, pour chaque section trouvée, les sept bits qui forment la période de base de sept jours. De cette façon, pour décrire entièrement une section, il suffit de conserver sa forme de base, son jour de début et son jour de fin.

**Algorithme A2:**

ETAPE 0: mlong := 2 (nombre maximal de semaines pour une exception)

mex := 2 (nombre maximal d'exceptions dans une semaine )

section := 0

Avancer jusqu'à la position p du premier bit allumé de $\delta_x$

ETAPE 1: "Recherche des sections"

section := section + 1

p1 := p

$\bar{\delta}_x^p := 0 \quad \forall p \in P$

Pour i:=p1 à p1+6 faire $\bar{\delta}_x^i := 1$

$\bar{\delta}_x := \bar{\delta}_x \wedge \delta_x$

forme[section] := $\bar{\delta}_x$ <<(p1-1)

$\delta_x' := 0$

Tantque( $||\delta_x'|| = 0$ et p<finperiode ) faire

$\bar{\delta}_x >> 7$

$\delta_x' := \bar{\delta}_x \bullet \delta_x$

p := p+7

Fin (Tantque)

Si (p>finperiode) p := finperiode

debut[section] := p1

fin[section] := p

```
                longueur[section] := p-p1
                Si (p < finperiode) Retourner à l'ETAPE 1

ETAPE 2:        "Unification"
                Trouver s tel que longueur[s] = max { longueur[i] }
                                                i=1..section
                Si (longueur[s] = 0) Terminer
                a) Tenter d'unifier la section s avec celles qui la précèdent:
                     next := s-1
                     Tantque (next>0  et longueur[next]>0) Faire
                          diff := forme[s] • forme[next]
                          Si
                          (||diff||=0 ou (||diff||≤mex et longueur[next]≤mlong))
                          Alors
                               Réunir les sections s et next
                               Mettre debut[next], fin[next] et  diff  dans  la
                               liste des cas d'exception de la section s.
                               longueur[next] := 0
                               next := next-1
                          Sinon
                               next := 0
                          Fin (Si)
                     Fin (Tantque)
                b) Tenter d'unifier la section s avec celles qui lui succèdent:
                     next := s+1
                     [Etape similaire a l'ETAPE 2 a), en incrémentant next]
                longueur[s] := 0
                Retourner à l'ETAPE 2
```

Il est intéressant de faire le lien entre cet algorithme et les méthodes alternatives de représentation des horaires de vols que nous avons décrites au début du chapitre trois. On remarque en effet que la division en sections obtenue après l'ETAPE 1 correspond en fait à une subdivision de l'horaire d'un vol en une série de semaines types consécutives. De même, le résultat obtenu après l'ETAPE 2 correspond à une représentation avec semaines types et listes de cas d'exception.

Appliquée ainsi, la méthode comporte encore une légère anomalie; les points de coupure entre les sections ne sont pas toujours faits aux meilleurs endroits. Reprenons notre exemple du début:

1100000110100011000001100000110000|0...0

En appliquant l'algorithme décrit, on obtient

JA01 FE04 67
AD JA11 ,

alors qu'on aurait préféré une description s'adaptant mieux aux dates civiles comme

JA01 JA31 67
AD JA11 .

Les deux descriptions sont exactes, mais la seconde respecte mieux les divicions traditionnelles du calendrier. Ainsi, plutôt que de toujours couper suivant des multiples de sept jours, on coupera selon les caractéristiques du calendrier lorsqu'un choix de coupures équivalentes se présentera. Pour y arriver, on crée un vecteur associant une priorité à chaque jour de la période. Lorsqu'un choix de points de coupure apparemment équivalents se présente, on décidera de couper après le jour qui a la plus haute priorité. Initialement, le vecteur des priorités peut, par

exemple, être défini ainsi:

- au départ, tous les jours ont une priorité nulle
- changements de saison: +30
- fin de mois : +15
- fin de semaine : +10
- jour 15 d'un mois : + 5

Ces choix sont arbitraires et ne constituent qu'un exemple. Ce sont des paramètres qu'un usager adaptera à ses besoins en y ajoutant des cas ou en modifiant la hiérarchie.

## 4.2 Organisation logique de la base de données

PLANAIR présente beaucoup d'affinités avec le logiciel EMME/2. Dans les deux cas, la grande quantité d'information à traiter et à laquelle on veut accéder de façon sélective, de même que l'aspect interactif-graphique qui permet l'accès aux données dans un ordre qui n'est pas prédéterminé, nécessitent l'utilisation d'une banque de données. Cependant, les deux systèmes divergent au niveau du type de traitement à effectuer sur les données de base. Après avoir décrit les principales ressemblances et différences entre les deux projets, nous passerons en revue les concepts fondamentaux qui ont présidé à l'organisation logique de la base de données de EMME/2 pour en évaluer l'applicabilité dans notre contexte.

Tel que mentionné plus tôt, EMME/2 est un système interactif-graphique pour la planification du transport urbain. Nous en schématisons ici les grandes étapes d'utilisation:

- L'entrée par lot ou de façon interactive des données de base. Ces dernières sont essentiellement composées d'un ensemble de modes de transport, d'un réseau de base comprenant les intersections, les liens et les centroïdes du réseau de transport considéré, des fonctions décrivant, entre autres, le taux de congestion des liens en fonction du flot y circulant et des matrices origines-destinations estimant la demande de transport entre les différents centroïdes du réseau.

- L'affectation des divers modes de transport sur le réseau. Ceci est fait à l'aide d'un modèle d'optimisation qui utilise les données de base en entrée. Les résultats sont stockés dans la banque de données même et constituent, avec le réseau, ce qui est appelé un scénario. Les calculs nécessaires à une affectation sont complexes et peuvent nécessiter quelques heures de traitement.

- L'analyse des résultats. Plusieurs programmes intégrés permettent une analyse complète des résultats, tant du point de vue graphique que numérique.

- Modifications des données de base. L'usager peut, de façon interactive, modifier les données de base pour ensuite effectuer une nouvelle affectation et pouvoir comparer les résultats obtenus avec les scénarios précédents. Ce cycle de modifications - affectation - comparaisons est ainsi répété plusieurs fois.

Pour fins de comparaisons, voici les grandes étapes d'utilisation de PLANAIR:

- L'entrée des données de base, par lot ou de façon interactive. Celles-ci sont en majeure partie composées d'un ensemble de véhicules disponibles, des noeuds desservis, des activités prévues à l'horaire et, possiblement, d'un ensemble d'accrochages obligatoires.

- Allocation des véhicules de la flotte aux activités. Ceci est réalisé en créant, modifiant et enlevant des accrochages à l'aide d'outils interactifs-graphiques utilisés conjointement avec des algorithmes d'allocation. Dans la dernière phase du projet, des outils analytiques en vue d'une optimisation globale seront aussi disponibles.

- Evaluation de l'horaire. Les résultats obtenus sont ensuite analysés par les différents départements impliqués et les corrections à apporter sont signalées au planificateur.

- Modifications de la solution en vue d'obtenir une nouvelle allocation. Il n'y a pas, ici, de véritable cycle de modifications - allocations - évaluation, puisque c'est la solution même qui est directement modifiée au fur et à mesure.

Comme on peut le voir, les deux schémas sont très similaires. Il existe toutefois une différence fondamentale entre eux: Emme/2 permet d'évaluer l'impact de modifications apportées au niveau des données de base, tandis que PLANAIR permet une véritable optimisation interactive. Ainsi, dans PLANAIR, modifications et allocations se confondent en une suite d'opérations effectuées de manière complètement interactive. L'usager qui effectue une modification peut voir immédiatement l'effet de celle-ci sur le reste de la solution et y réagir aussitôt. De cette façon,

les modifications apportées sont beaucoup plus nombreuses et plus fréquentes.

Voyons maintenant l'impact de ceci sur l'organisation de la base de données. Du point de vue machine, la base de données consiste en un seul fichier à accès aléatoire qui peut être adressable par mots, par enregistrements ou par octets, selon l'installation. Du point de vue des programmes, cependant, la base de données consiste en un ensemble de fichiers internes distincts, chacun pouvant être considéré comme un fichier à accès aléatoire et possédant ses propres caractéristiques comme le nombre d'enregistrements, le nombre de mots par enregistrements et le type de données (entier, réel ou texte). L'interface entre la représentation externe et interne est faite par un ensemble de routines d'accès-disque faisant partie de la bibliothèque de sous-routines de EMME/2.

Tel que décrit par Spiess (1984), la base de données de EMME/2 est organisée suivant ces cinq concepts fondamentaux:

1) Organisation verticale des fichiers internes

2) Données de base et résultats emmagasinés dans la banque de données

3) Absence de redondance

4) Détection des erreurs à l'entrée

5) Données de base stockées dans des tables ordonnées selon l'identificateur des éléments

Reprenons un à un ces concepts:

1) Organisation verticale des fichiers internes

Les fichiers internes contiennent des données structurées *verticalement*; plutôt que d'avoir un enregistrement pour chaque élément et contenant tous ses attributs (structure *horizontale*), il y a un enregistrement pour chaque attribut, contenant la valeur de cet attribut pour chaque élément. Notre banque de données, tout comme celle de EMME/2, est constituée de milliers d'éléments, chacun comportant peu d'attributs. De ce fait, la méthode d'organisation verticale réduit le nombre d'accès-disque nécessaires. De plus, une meilleure gestion de l'espace mémoire central peut être faite car il est possible de ne lire du disque que les attributs nécessaires.

Nous avons donc jugé opportun d'adopter aussi cette organisation, d'autant qu'elle rend plus efficace la sélection d'un sous-ensemble d'activités basée sur les caractéristiques de leurs attributs. C'est une operation fréquemment effectuée, qui nécessite que l'on parcoure tour à tour certains attributs des activités pour déterminer lesquelles satisfont aux critères de sélection. On peut être intéressé, par exemple, à travailler avec le sous-ensemble des activités partant du noeud i entre neuf heures et midi. L'organisation verticale permet de n'amener en mémoire centrale que les enregistrements concernant les noeuds et les heures de départ des activités pour déterminer le sous-ensemble visé. Une organisation horizontale obligerait, elle, à amener en mémoire centrale tous les attributs de chaque activité, ce qui occuperait beaucoup plus d'espace mémoire central que nécessaire. Enfin, l'organisation verticale ne comporte que des enregistrements composés d'éléments du même type, ce qui évite l'emploi de structures composées.

Le détail de l'organisation des fichiers internes est donné dans l'appendice A.

2) Données de base et résultats stockés dans la banque

EMME/2, qui constitue un environnement de travail centré sur un modèle d'affectation, garde dans la banque les données de base (réseau, fonctions, etc.) de même que les résultats obtenus du modèle (affectation). Cela permet d'utiliser ces résultats comme données de base pour une étape subséquente d'analyse ou de prévision.

Dans notre cas, du fait que les modifications sont appliquées directement aux résultats, et comme modifications et allocations sont des opérations qui se chevauchent, la frontière entre données de base et résultats est beaucoup plus floue. Dans un tel contexte plus dynamique, il est donc clair que données de base et résultats doivent faire partie intégrante de la banque de données. Ainsi, tous les accrochages générés par les différents algorithmes d'allocation, ou entrés interactivement par l'usager, seront intégrés à la banque de données.

3) Absence de redondance

Afin d'éviter les problèmes d'inconsistance de la banque de données, une politique de non-redondance des données stockées a été strictement suivie lors de la conception de EMME/2. Là où il est raisonnable de le faire, les fichiers internes sont définis de manière à éviter les informations redondantes sur disque. Ceci implique que chaque vecteur de données secondaires doit être recalculé chaque fois qu'il est utilisé. Les coûts entraînés par ces calculs sont cependant amplement justifiés par le fait qu'une modification au niveau des données primaires est implicitement faite pour toutes les données qui en sont dérivées. On ne risque pas ainsi qu'une modification ait des répercussions en cascade sur d'autres données

secondaires ou de plus haut niveau.

Nous sommes aussi d'avis que, pour une banque de données appelée à être modifiée fréquemment de façon interactive, c'est la seule méthode vraiment efficace pour s'assurer de la consistance des informations qui y sont contenues. De plus, si on envisage l'utilisation d'une politique de détection des erreurs à l'entrée, l'absence de redondance allège le travail nécessaire pour valider et effectuer les modifications.

4) Détection des erreurs à l'entrée

Pour pouvoir garder la banque de données dans un état consistant et assurer l'intégrité de son contenu, toute modification apportée à cette dernière n'est effectuée qu'après avoir vérifié la validité de l'opération et s'être assuré qu'elle ne causera aucune inconsistance. Ce n'est qu'une fois ces tests réalisés que la modification sera apportée. Définissant l'état initial vide comme un état consistant, cette règle simple mais efficace permet de garantir que les transitions ne se font que d'un état consistant à un autre état consistant. Ainsi, un programme voulant utiliser la banque de données est assuré de la trouver en bon état et il est de sa responsabilité de veiller à ce que chaque modification qu'il désire apporter soit permise, de façon à retourner la banque dans un état consistant.

Cette politique restreint les actions possibles en ne permettant d'effectuer qu'une seule modification à la fois. Mais, dans un contexte interactif, est-il vraiment souhaitable de permettre à un usager d'effectuer plusieurs modifications simultanées? Le risque est grand, alors, que les modifications provoquent des effets de bord les unes sur les

autres et entraînent des effets imprévus. Et si une erreur est détectée, il faut déterminer par laquelle (ou lesquelles) des modifications cette erreur a été causée, ce qui peut être assez difficile. Ainsi, contrairement à ce qui peut paraître au premier abord, le fait de limiter les modifications à une seule à la fois ne constitue pas un point négatif pour l'usager. Ce dernier peut alors toujours bien évaluer la portée de ses actions, et si une erreur survient, il sait automatiquement que c'est l'unique modification qu'il vient d'effectuer qui est en cause et nulle autre.

Pour ces raisons, nous estimons que *détection des erreurs à l'entrée* et *absence de redondance* vont de pair et sont à la base d'un système interactif simple et fiable.

5) Données de base stockées dans des tables ordonnées

L'opération la plus fréquente pour EMME/2 est celle du parcours systématique d'un ensemble d'éléments. Il y a peu de modifications comparativement aux accès, et ces derniers sont, le plus souvent, systématiques. Pour cette raison, les données de base sont, pour la plupart, stockées dans des tables séquentielles ordonnées, triées selon leur identificateur. En plus d'être la structure la plus simple et la plus efficace pour des parcours systématiques, la table séquentielle ordonnée comporte plusieurs avantages:

- Une recherche dichotomique peut être directement utilisée lorsqu'il est nécessaire d'accéder à un élément particulier.

- Les éléments étant déjà ordonnés selon leur identificateur, il n'est pas nécessaire de les trier pour produire des sorties.

- Deux structures qui contiennent les mêmes éléments sont physiquement identiques, peu importe l'histoire de leur développement.

- Il est possible d'utiliser une lecture par tampons lorsque l'espace mémoire central est limité.

- Cette organisation simple permet de comparer facilement deux structures complexes en les parcourant en parallèle.

Cette structure, toute indiquée pour EMME/2, comporte cependant un grand désavantage dans notre contexte où l'aspect modification occupe une place prépondérante: il est difficile et inefficace d'y effectuer les mises à jour. De plus, dans notre cas, l'accès aux données est beaucoup moins systématique car il dépend plus directement des demandes de l'usager. L'avantage majeur de la table séquentielle est donc perdu.

Il est par conséquent essentiel d'utiliser une structure interne plus dynamique qui se prête mieux à des modifications fréquentes. C'est pour ces raisons que nous avons choisi d'utiliser des listes chaînées pour stocker plusieurs données de base; ce sont des structures très simples et la mise à jour y est une opération immédiate. On perd plusieurs des avantages secondaires de la table séquentielle, mais la flexibilité gagnée en vaut le coup.

Il est fréquemment nécessaire d'accéder à une activité précise. La recherche dichotomique directe n'étant plus possible, une table

séquentielle de pointeurs, ordonnés selon l'identificateur de l'activité vers laquelle ils pointent, est gardée en parallèle. Il devient de la sorte possible, avec une indirection, d'effectuer une recherche dichotomique et d'obtenir une sortie triée des activités.

Un autre point concerne la gestion de l'espace mémoire central: l'accès aux données n'étant pas effectué selon un parcours systématique et celles-ci étant organisées en structures de listes, il n'est pas possible d'employer une lecture par tampons consécutifs. Cependant, un usager travaillera typiquement avec un sous-ensemble restreint de données (les activités réalisées avec une certaine classe de véhicules, les événements se déroulant à un noeud donné, etc.). Il s'ensuit qu'une antémémoire ("cache memory") pourra être utilisée avec profit. Une recherche dans ce tampon situé en mémoire centrale déterminera si l'information désirée s'y trouve et, dans le cas contraire, un des éléments qui y est contenu pourra être retourné sur disque pour pouvoir amener l'élément voulu. Il découlera de l'utilisation de cette antémémoire une meilleure utilisation de l'espace mémoire central et un gain appréciable sur les temps d'accès-disque. Une première version de la routine de gestion de ce tampon mémoire a été implantée. De nombreux raffinements doivent encore y être apportés, mais déjà, la réduction des temps d'exécution est fort appréciable.

Signalons enfin que l'utilisation de structures dynamiques ne permettra plus que deux banques qui contiennent les mêmes éléments soient physiquement identiques, indépendamment de l'histoire de leur développement. De même, il devient beaucoup plus difficile de comparer deux ensembles de données, dû à leur organisation interne plus complexe. Nous sommes conscients du surcroît de travail qu'il faudra effectuer pour réaliser ces

opérations lorsqu'il sera nécessaire de comparer deux horaires alternatifs mais, globalement, les avantages d'une structure dynamique l'emportent quand même sur les inconvénients.

## 4.3 Structures de données

Au cours de la section précédente, il a été question de l'organisation logique générale de la banque de données. Dans la présente section, nous traiterons de façon plus détaillée des structures de données utilisées.

La banque de données de PLANAIR comporte quatre principaux groupes de données:

- les véhicules
- les noeuds
- les activités
- les accrochages

Ces groupes possèdent des caractéristiques qui leur sont propres et forment une hiérarchie qu'il est nécessaire de respecter lors de l'entrée des données dans la banque. Ainsi, avant d'établir un accrochage, il faut définir les activités concernées qui, à leur tour, demandent que soient définis les noeuds desservis et les classes de véhicules utilisées. De même, certaines caractéristiques des noeuds ont trait aux types de véhicules qui peuvent y être accueillis. Il est donc nécessaire que ces derniers soient préalablement définis. La figure 4-1 schématise les relations existant entre les groupes et donne les principaux attributs de ceux-ci.

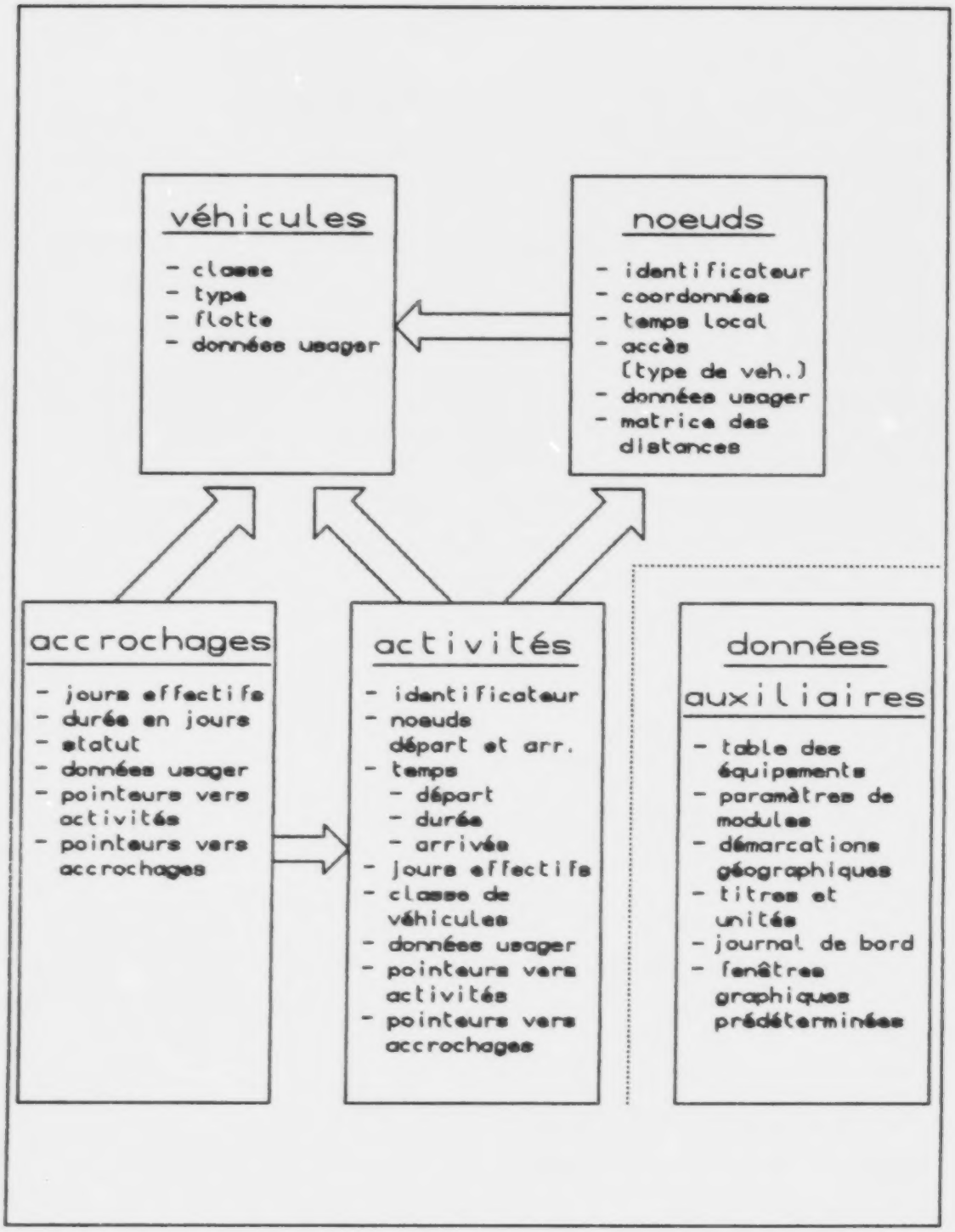


FIGURE 4-1: Contenu de la banque de données de PLANAIR.

La flotte des véhicules disponibles est un élément relativement stable; peu de changements seront apportés à ces données. Nous avons donc opté pour l'utilisation d'une table séquentielle triée selon l'ordre alphabétique des identificateurs des classes de véhicules. Cette structure simple est plus statique, mais permet un accès rapide aux données concernant la flotte des véhicules. L'identificateur des classes de véhicules est constitué de quatre caractères alphanumériques. Il doit être unique, c'est-à-dire que deux classes ne peuvent avoir le même.
La table des véhicules peut contenir jusqu'à MVCLAS classes de véhicules. Chacune possède un attribut qui indique à quel type de véhicule elle appartient. MVTYP étiquettes alphanumériques sont disponibles à cette fin.

Les noeuds constituent aussi des éléments très stables de la banque de données. Ici encore, une table séquentielle est utilisée pour représenter ces données. Organisée selon l'ordre alphabétique des identificateurs des noeuds, celle-ci permet une recherche dichotomique pour trouver rapidement les informations rattachées à un noeud spécifique. Des modifications telles que retraits ou ajouts nécessitent plus de travail, mais ces dernières n'étant effectuées que rarement, l'inconvénient est mineur. L'identificateur des noeuds est aussi composé de quatre caractères alphanumériques et doit être différent pour chaque noeud.

La table des noeuds peut contenir jusqu'à MNODE définitions. Chaque noeud comporte des attributs qui concernent les types de véhicules qui peuvent y être accueillis. Ces attributs font donc référence à la table des véhicules définie précédemment. De plus, pour chaque noeud, un pointeur (IFACT) indique la première activité à avoir lieu à celui-ci (chronologie horaire). Une valeur négative de IFACT indique que cette première activité

part du noeud ($\alpha \varepsilon A_i^+$), une valeur positive, que l'activité s'y termine ($\beta \varepsilon A_i^-$), et une valeur nulle indique l'absence d'activité. Evidemment, lorsqu'on crée la table des noeuds, aucune activité n'est encore définie et, par conséquent, ce pointeur est nul pour tous les noeuds.

Les **activités**, avec les accrochages dont il sera question plus loin, constituent la partie dynamique de la banque de données. Comme mentionné plus tôt, nous avons choisi d'utiliser une structure de liste pour stocker ces données qui seront fréquemment modifiées. Les activités sont conservées dans une liste, selon un ordre quelconque, suivant les disponibilités de la liste d'espaces libres. La table des activités peut contenir jusqu'à MACT activités.

Dans le domaine aérien, la notion d'identificateur ou de numéro de vol, est large et s'applique à un ensemble d'activités. Par conséquent, les règles concernant l'attribution des identificateurs aux activités posent un certain problème: doit-on exiger que chaque activité ait un identificateur propre, différent de tous les autres? Si l'on applique cette règle, il faut alors que l'usager donne un identificateur différent à tous les segments qui composent un vol, utilisant possiblement un suffixe pour reconnaître les segments d'un même vol (129a, 129b, 129c, par exemple). Il devient aussi nécessaire d'attribuer un identificateur différent à tous les segments de vol dont l'heure de départ ou d'arrivée est modifiée à cause des changements horaires saisonniers et qui, conséquemment, deviennent des activités différentes. Si l'on multiplie le nombre de changements horaires saisonniers par le nombre de segments pouvant composer un vol, on se retrouve vite avec une profusion d'identificateurs pour un seul vol. On impose alors à l'usager la tâche supplémentaire de se souvenir à quoi

correspond chaque identificateur d'un même vol.

Si, par contre, on permet que plusieurs activités partagent le même identificateur, il devient alors nécessaire, pour avoir accès à une activité particulière, de la distinguer en donnant la valeur de quelques uns de ses attributs, ce qui peut alourdir le dialogue. Il faudra, par exemple, spécifier que l'on veut l'activité 129 qui débute au noeud YUL. A ce moment, on n'a plus vraiment d'identificateur; il s'agit alors d'un nom collectif qui devient un moyen de regrouper des activités ayant un point commun pour l'usager. De plus, n'obligeant plus l'usager à spécifier une activité unique au cours des différents dialogues avec le système, le travail de programmation s'en trouve accru, puisque le système doit toujours pouvoir travailler à partir d'ensembles d'activités. Par exemple si, au cours d'un dialogue, l'usager indique qu'il désire créer un accrochage entre les activités 129 et 130, le système doit pouvoir déduire quelles sont les activités compatibles parmi ces deux ensembles, car il ne s'agit plus simplement de créer un accrochage entre une activité 129 et une activité 130. Par contre, on met à la disposition de l'usager habitué à manier les ensembles, un outil beaucoup plus puissant et flexible. Nous sommes d'avis que le fait d'exiger que chaque activité ait son identificateur propre serait d'imposer à l'usager une discipline superflue. Comme une des règles de base dans PLANAIR est de laisser à l'usager le plus de latitude possible, nous avons opté pour la seconde approche. Si un usager ne se sent pas à l'aise avec l'idée d'un ensemble d'activités regroupées sous un même identificateur propre, libre à lui de les définir de façon unique. Nous ne voulons pas imposer à l'usager une méthode de travail rigide sous le seul prétexte d'une plus grande facilité de programmation. Dans ce qui suit, nous utiliserons quand même le terme *identificateur* pour

désigner les quatre caractères alphanumériques dont l'usager dispose pour nommer les activités, bien qu'il n'y ait pas de notion d'unicité rattachée à ceux-ci.

Pour l'accès au groupe d'activités de même identificateur ou la production de rapports, un tableau de pointeurs ordonnés selon l'identificateur des activités est construit et conservé. Toutefois, cet accès aux activités selon l'ordre alphabétique de leur identificateur n'est pas suffisant. Il est souvent nécessaire d'accéder aux activités suivant leur déroulement dans le temps et l'espace pour obtenir l'achalandage d'un noeud, par exemple, ou encore la liste des événements candidats pour un accrochage. A cette fin, des pointeurs sont utilisés pour ordonner les activités en structures de listes, selon la chronologie horaire et selon leur noeud de départ ou d'arrivée. Les pointeurs IFACT de chacun des noeuds constituent les têtes de ces listes. Tous les pointeurs subséquents dans ces listes suivent la même convention que IFACT: une valeur positive indique une arrivée, une valeur négative, un départ et une valeur nulle, la fin de la liste. Chaque activité possède deux variables (INODE et JNODE) indiquant le noeud de départ et d'arrivée de celle-ci, et deux pointeurs (INEXT et JNEXT) qui pointent respectivement vers la prochaine activité à se dérouler à INODE et à JNODE. La figure 4-2 illustre la structure de listes entrecroisées obtenue. De cette façon, chaque activité fait partie de deux listes: la liste de son noeud de départ (INODE) où elle est insérée selon son heure de départ, et celle de son noeud d'arrivée (JNODE), selon son heure d'arrivée.

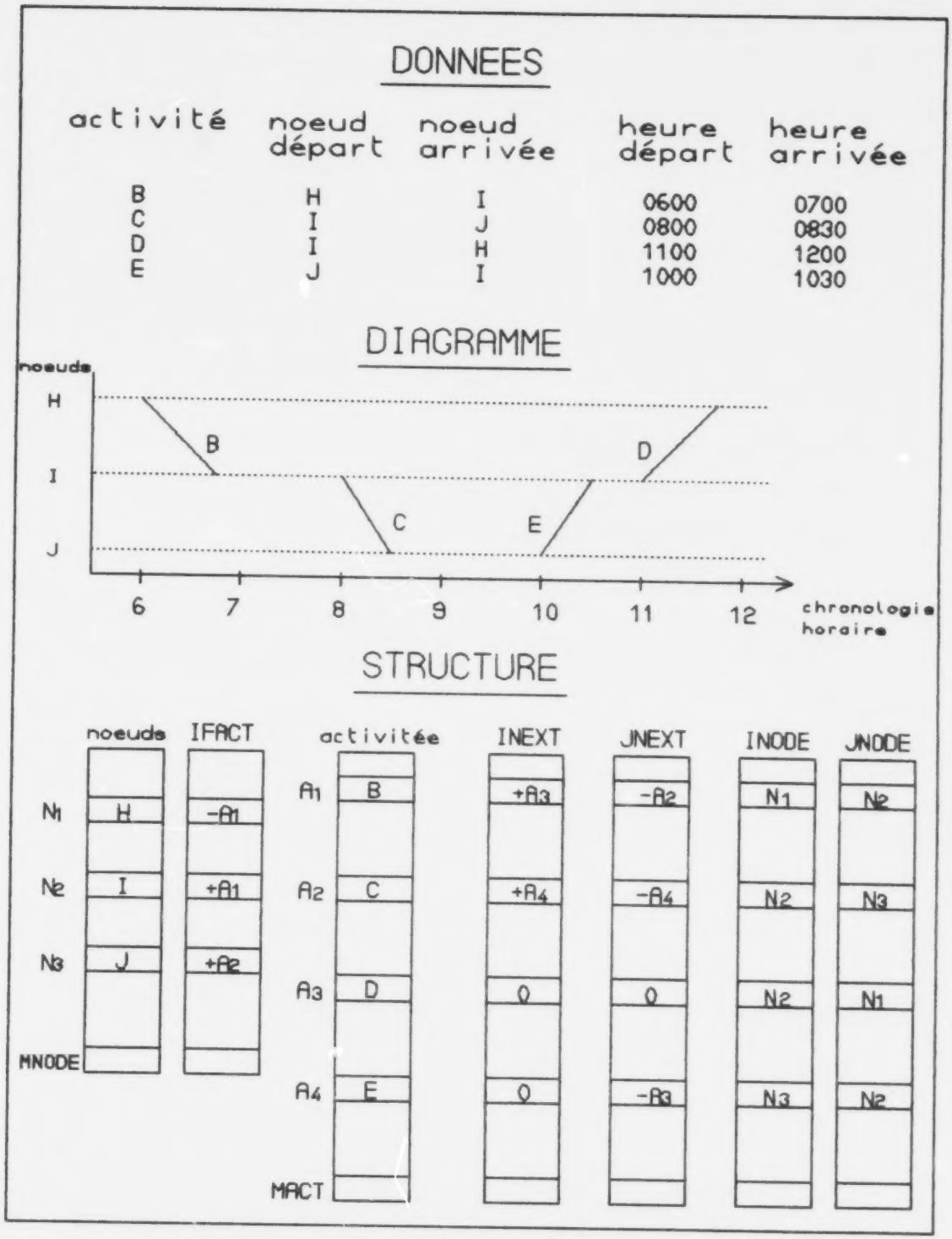


FIGURE 4-2: Structure des pointeurs des activités.

Il faut bien noter que cette structure est organisée selon la chronologie horaire des activités, et non selon la chronologie réelle des événements; c'est-à-dire que le facteur d'ordonnancement est l'heure de départ ou d'arrivée de l'activité, sans tenir compte des dates où cela doit se produire. Il ne s'agit donc pas d'une chronologie stricte. C'est là une nuance importante. A première vue, il peut sembler plus logique, et surtout plus utile, d'ordonner ces listes de manière plus conforme à la réalité, suivant l'ordre des événements qui surviennent à un noeud. Trois points nous ont cependant amenés à la conclusion contraire:

a) Comme nous l'avons vu au chapitre précédent, une chronologie horaire nous permettra de travailler uniquement au niveau des activités et de réaliser un certain parallélisme de calcul. C'est donc à partir de ces listes que nous pourrons implanter les fonctions ARR et DEP décrites à la section 3.4.

b) Une organisation selon la chronologie des événements nécessiterait un travail de mise à jour énorme suite à une modification au niveau d'une activité puisque l'effet se répercuterait sur tous les événements qui la composent.

c) Utilisant conjointement les vecteurs de bits représentant les événements et une structure organisée selon la chronologie des activités, il est possible d'obtenir la chronologie des événements, sans trop d'efforts.

L'algorithme suivant, qui parcourt toutes les activités survenant à un noeud $i \in N$, illustre l'utilisation de cette structure de données.

**Algorithme B1:**

ETAPE 0: ipoint := IFACT(i) , $i \varepsilon N$

ETAPE 1: Si (ipoint = 0) alors Terminer

ETAPE 2: $\alpha$ := |ipoint|
Traitement de l'activité $\alpha$
(Si ipoint<0, $\alpha \varepsilon A_i^+$ , sinon $\alpha \varepsilon A_i^-$)

ETAPE 3: Si (ipoint<0) alors ipoint=inext(-ipoint)
sinon ipoint=jnext(ipoint)
Retourner à l'ETAPE 1.

C'est un algorithme de base très simple. Si l'on veut obtenir l'ordre chronologique des événements au noeud i, à partir du jour j dans la période, il n'y a qu'à le modifier ainsi:

**Algorithme B2:**

ETAPE 0: ipoint := IFACT(i) , $i \varepsilon N$
p := j , $j \varepsilon P$

ETAPE 1: Si (ipoint = 0) alors Terminer

ETAPE 2: $\alpha$ := |ipoint|
Si ($\delta_\alpha^p \neq 0$) alors Traiter e=($\alpha$,p)

ETAPE 3: Si (ipoint<0) alors ipoint:=inext(-ipoint)
sinon ipoint:=jnext(ipoint)
Si (ipoint $\neq$ 0) alors Retourner à l'ETAPE 2

ETAPE 4: p := p+1
Si ($p \varepsilon P$) alors
ipoint := IFACT(i)
Retourner à l'ETAPE 2
Terminer.

Les **accrochages** constituent la partie prépondérante des données. Ce sont des éléments auxquels on aura fréquemment accès et qui seront appelés à être modifiés très souvent. Tout comme les activités, les accrochages

sont conservés dans une liste, selon un ordre quelconque, suivant les disponibilités de la liste d'espaces libres. Cette liste peut contenir jusqu'à MTURN accrochages. Contrairement aux activités cependant, les accrochages n'ont pas d'identificateur propre. Ceux-ci sont en effet identifiés par les activités qu'ils relient. Chaque accrochage $\tau=(\alpha,\beta,\lambda_\tau)$ a donc deux pointeurs (IENDA et IBEGA) pointant respectivement vers $\alpha$ et $\beta$.

Il est particulièrement important de retracer les ensembles d'accrochages prédécesseurs ($T_\alpha^-$) et successeurs ($T_\alpha^+$) d'une activité $\alpha$. En effet, il sera fréquent de vouloir vérifier si une activité est saturée et, si elle ne l'est pas, d'identifier les événements qui ne sont pas accrochés. Il est donc nécessaire d'établir une structure qui permette un accès rapide aux ensembles $T_\alpha^+$ et $T_\alpha^-$, mais qui soit aussi flexible pour des modifications fréquentes. Des pointeurs établissant deux listes entrecroisées d'accrochages sont utilisés pour retracer $T_\alpha^-$ et $T_\alpha^+$. Chaque activité $\alpha$ utilise deux variables (ITURN et JTURN) qui pointent respectivement vers un premier accrochage prédécesseur et un premier accrochage successeur. Chaque accrochage $\tau=(\alpha,\beta,\lambda_\tau)$ possède à son tour deux pointeurs permettant d'accéder aux accrochages suivants dans la liste: NEXEND qui pointe vers un accrochage $\tau=(\gamma,\alpha,\lambda_\tau)$ , $\gamma\varepsilon A_i^-$ et NEXBEG qui pointe vers un accrochage $\tau=(\alpha,\chi,\lambda_\tau)$ , $\chi\varepsilon A_i^+$. (Les fins de liste sont dénotées par des pointeurs nuls.) Chaque accrochage $\tau=(\alpha,\beta,\lambda_\tau)$ fait ainsi partie de deux listes: une liste vers l'avant à partir de JTURN($\alpha$) et une liste vers l'arrière à partir de ITURN($\beta$). La figure 4-3 illustre ces structures. Règle générale, pour chaque activité, ces listes sont très courtes, comportant moins de dix accrochages. Cela permet un temps d'accès rapide à tous les éléments, même pour le dernier qui nécessite un parcours de toute la liste pour y arriver.

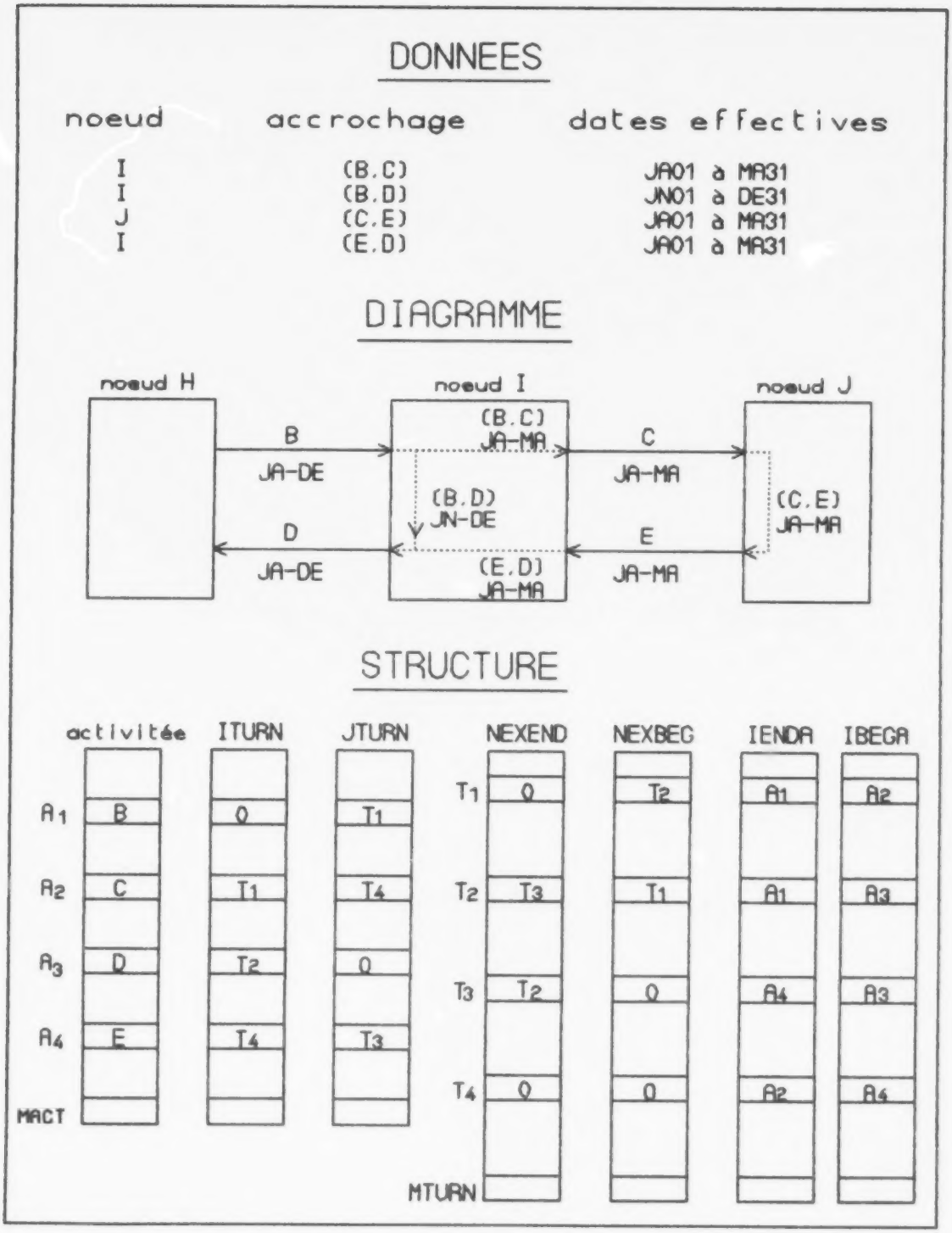


FIGURE 4-3: Structure des pointeurs des accrochages.

Les algorithmes suivants, qui vérifient si une activité $\alpha$ est saturée, montrent comment on peut se servir de ces listes.

**Algorithme B3:** Vérifier si $\alpha$ est post-saturée

ETAPE 0: $\tau$:=JTURN($\alpha$)

$\bar{\delta}_\alpha := \delta_\alpha >> \lambda_\alpha$

ETAPE 1: Tantque ($\tau \neq 0$) faire

$\bar{\delta}_\alpha := \bar{\delta}_\alpha \bullet \delta_\tau$

$\tau$ := NEXBEG($\tau$)

Fin (Tantque)

$\delta'_\alpha := \bar{\delta}_\alpha << \lambda_\alpha$

Si ($||\delta'_\alpha|| = 0$) alors

$\alpha$ est post-saturée

Sinon

$\delta'_\alpha$ contient les événements de $\alpha$ qui ne sont pas post-accrochés.

Fin (Si)

L'algorithme suivant est très similaire, étant le symétrique du précédent.

**Algorithme B4:** Vérifier si $\alpha$ est pré-saturée

ETAPE 0: $\tau$:=ITURN($\alpha$)

$\bar{\delta}_\alpha := \delta_\alpha$

ETAPE 1: Tantque ($\tau \neq 0$) faire

$\bar{\delta}_\alpha := \bar{\delta}_\alpha \oplus \{\delta_\tau >> \lambda_\tau\}$

$\tau$ := NEXEND($\tau$)

Fin (Tantque)

$\delta'_\alpha := \bar{\delta}_\alpha$

Si ($||\delta'_\alpha|| = 0$) alors

$\alpha$ est pré-saturée

Sinon

$\delta'_\alpha$ contient les événements de $\alpha$ qui ne sont pas pré-accrochés.

Fin (Si)

Utilisant ces deux algorithmes et les structures que nous venons de décrire, il nous est maintenant possible de donner plus de détails concernant l'implantation de l'algorithme développé à la section 3.4 . Nous approfondirons en particulier les détails liés à l'implantation des fonctions ARR et DEP à partir des listes de chronologie horaire des activités.

L'algorithme comportera maintenant deux phases. Au cours de la première, nous créerons tous les accrochages possibles avec $\lambda_\tau = 0$ et, simultanément, bâtirons les listes IARR et IDEP qui remplaceront les fonctions ARR et DEP dans la seconde phase. Pour ce faire, nous parcourrons la liste de chronologie horaire en la "séparant", un peu à la manière d'une fermeture éclair, et en réorganisant les pointeurs. La figure 4-4 schématise le procédé.

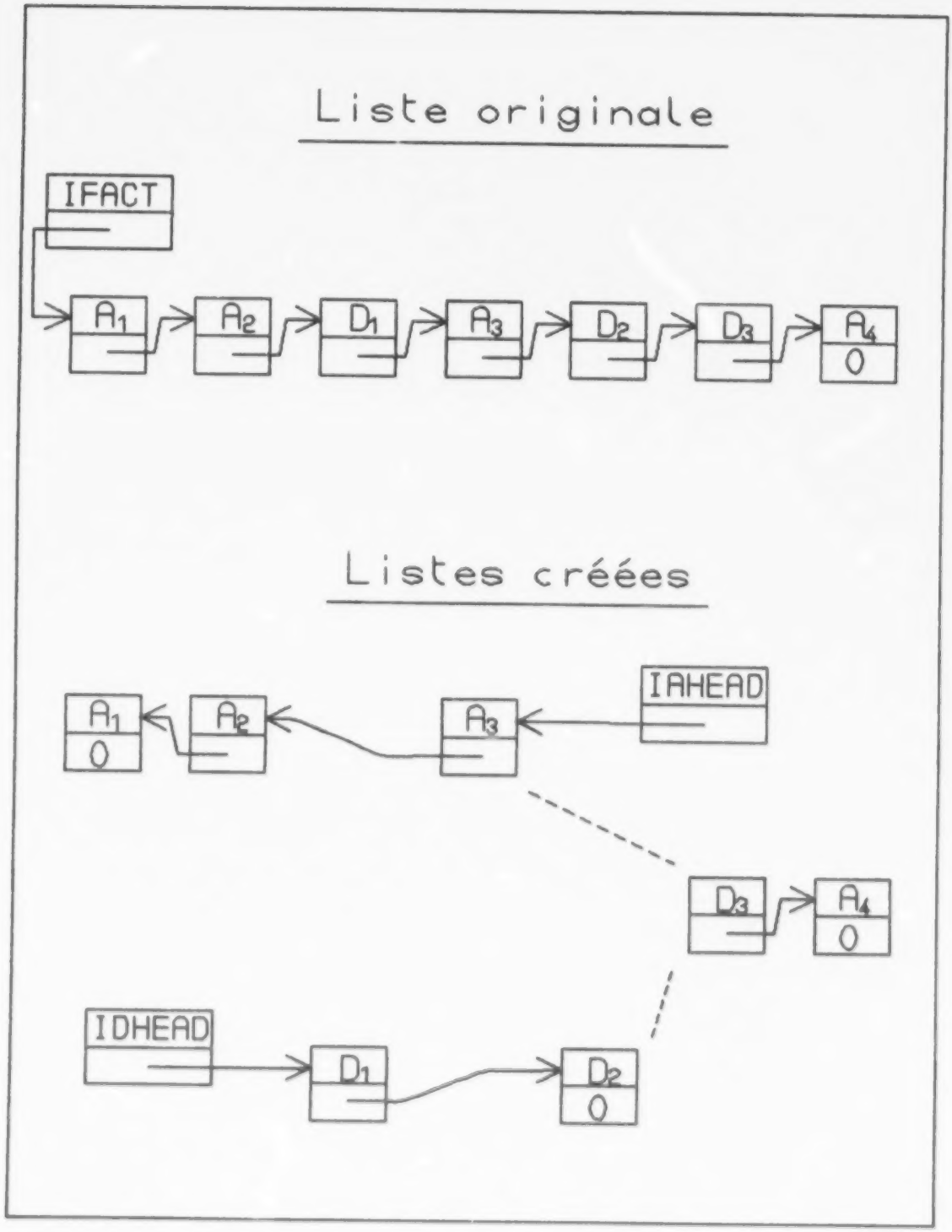


FIGURE 4-4: Listes créées au cours de la phase I de l'algorithme de génération des accrochages. Les $A_i$ représentent des arrivées, et les $D_i$ des départs.

La seconde phase génèrera les accrochages avec $\lambda_T = 1 \ldots$ MAX. Nous noterons par IAHEAD et IDHEAD les têtes des listes IARR et IDEP. Voici l'algorithme obtenu en incorporant ces détails à l'algorithme A1:

**Algorithme B5:**

ETAPE 0: $\lambda := 0$

isucc := IFACT(i)

Si (isucc = 0) Terminer (Aucune activité à ce noeud.)

ETAPE 1: "Fermeture éclair"

Si (isucc = 0) Aller à l'ETAPE 3

Si (isucc > 0) alors

$\alpha$ := isucc

isucc := JNEXT(isucc)

Appliquer B3 à $\alpha$

Si ($\left|\left|\delta'_\alpha\right|\right| \neq 0$) alors

($\alpha$ n'est pas déjà post-saturée)

Mettre $\alpha$ et $\delta'_\alpha$ au début de la liste IARR

Fin (Si)

Retourner à l'ETAPE 1

Sinon

$\beta := -$isucc

isucc := INEXT(-isucc)

Appliquer B4 à $\beta$

Si ($\left|\left|\delta'_\beta\right|\right| \neq 0$) alors

($\beta$ n'est pas déjà pré-saturée)

Mettre $\beta$ et $\delta'_\beta$ à la fin de la liste IDEP

Sinon

Retourner à l'ETAPE 1.

Fin (Si)

Fin (Si)

ETAPE 2: Essayer de pré-saturer $\beta$

$\alpha$ := IAHEAD (i.e. $\alpha := ARR^{-}(\beta)$ )

PRESATURER($\beta, \delta'_{\beta}, \lambda$)

Si (isucc = 0) Aller à l'ETAPE 3

Retourner à l'ETAPE 1.

ETAPE 3: Phase II: Création des accrochages avec $\lambda_T > 0$

$\lambda := 0$

$\beta$ := IDHEAD

Tantque ($\lambda <$ MAX et $\beta \neq 0$) faire

$\lambda := \lambda + 1$

$\delta'_{\beta} := \delta'_{\beta} >> 1$

PRESATURER($\beta, \delta'_{\beta}, \lambda$)

$\beta$ := IDEP($\beta$)

Fin (Tantque)

La routine PRESATURER est en fait la boucle interne de l'algorithme A1 modifiée pour introduire les listes IARR et IDEP à la place des fonctions ARR et DEP:

**Fonction PRESATURER:**

ETAPE 0: $\alpha$ := IAHEAD($\beta$)

ETAPE 1: Tantque ( $\alpha \neq 0$ et $||\delta'_\beta|| \neq 0$) Faire

$\bar{\delta}_\alpha := (\delta'_\alpha >> \lambda_\alpha)$

Si ($v^-_\alpha = v^+_\beta$ et $||\bar{\delta}_\alpha \wedge \delta'_\beta|| \neq 0$) Alors

a) Créer accrochage $\tau = (\alpha, \beta, \lambda)$

b) Réduire le problème

Fin (Si)

$\alpha$ := IARR($\alpha$)

Fin (Tantque)

Les listes IARR et IDEP sont des sous-produits particulièrement utiles; tous les éléments qui restent dans ces listes sont ceux qui n'ont pu être saturés. En parcourant ces listes, on obtient donc immédiatement tous les cas "intéressants" qui posent des problèmes. Trois raisons font qu'il peut rester des activités non saturées:

i) Comme nous l'avons vu à la section 4.1, le début et la fin de la période représentent des coupures artificielles dans l'horaire des activités. On doit donc s'attendre à ce que les premiers et les derniers événements à survenir à un noeud ne puissent être accrochés. Pour y arriver, il faudrait déborder dans les périodes adjacentes à celle qui est à l'étude.

ii) La valeur fixée de MAX n'était pas assez grande. Dans ce cas, les activités restantes pourraient possiblement être saturées, mais en créant de accrochages de durée plus longue. En choisissant volontairement une petite valeur de MAX, on a la possibilité de générer un

horaire qui comportera plusieurs grands "trous" où les activités ne seront pas accrochées. On peut ensuite utiliser ces trous pour intégrer à l'horaire les périodes d'entretien qui exigent qu'un véhicule demeure longtemps au sol.

iii) L'horaire considéré n'est pas balancé; le nombre d'arrivées et de départs est inégal. Dans ce cas, l'usager doit prendre des mesures correctives au niveau de l'horaire des activités.

Plusieurs améliorations peuvent encore être apportées à cet algorithme lors de la programmation. Deux points en particulier méritent d'être mentionnés:

- Puisque le problème est séparable par classe de véhicules, on peut, à l'ETAPE 1 de B5, construire une liste IARR et une liste IDEP pour chaque classe de véhicules rencontrée. Ceci permet de diminuer sensiblement le travail nécessaire lors de la recherche d'une activité compatible dans la routine PRESATURER, en restreignant les activités examinées au sous-ensemble de celles qui sont de la même classe de véhicules.

- Dans l'algorithme que nous avons décrit, la variable MAX ne peut être définie qu'en terme de jours. En apportant une modification très simple, on peut définir cette variable en termes d'heures et de minutes, ce qui permet un contrôle beaucoup plus fin sur la durée maximale permise des accrochages. De même, en ajoutant une variable MIN, on peut contrôler la durée minimale permise pour les accrochages.

Afin d'effectuer des tests sur les structures de données et les algorithmes développés, nous avons créé une banque de données composée d'un

horaire réel complet de la société Air Canada. L'horaire couvre une période d'un an, d'octobre 1983 à octobre 1984. La banque de données résultante comprend principalement:

- 114 noeuds
- 14 classes de véhicules
- plus de 2000 activités (soit au-delà de 314 000 événements)
- plus de 3800 accrochages

La taille totale de cette banque est inférieure à un méga-octet d'espace disque.

L'algorithme LIFO de génération des accrochages a été appliqué avec des valeurs extrêmes sur la durée des accrochages (MIN=0, MAX=7 jours). Pour générer les accrochages à tous les noeuds, pour toutes les classes de véhicules et toutes les activités sur l'ensemble de la période, le temps de calcul requis est d'environ 8 minutes (temps réel) sur un micro-ordinateur Pixel 100/AP. Il est intéressant de constater que près de la moitié de ce temps est utilisé pour traiter seulement l'aéroport de Toronto qui est, de loin, l'aéroport de base le plus important pour Air Canada. Le graphique de la figure 4-5 donne un aperçu de la relation entre la taille du sous-problème traité et le temps de calcul requis, pour chaque aéroport. On remarque que, pour plus de 85% des sous-problèmes, le temps de calcul est inférieur à 3 secondes, ce qui représente un temps de réponse presqu'immédiat pour un usager qui travaille interactivement. On peut aussi voir que l'algorithme utilisé est stable, le temps de calcul étant lié de façon presque linéaire à la taille du sous-problème, même pour les aéroports où la circulation est intense.

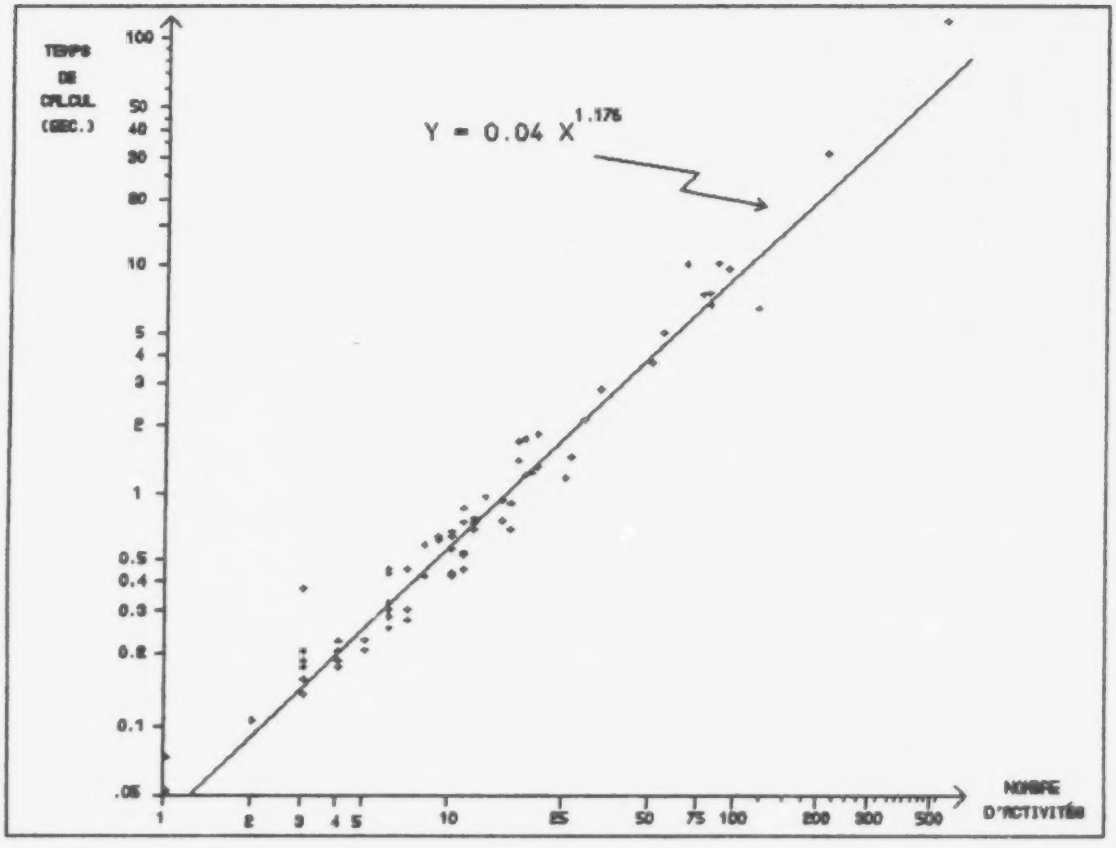


FIGURE 4-5: Résultats obtenus avec l'algorithme de génération des accrochages. Les échelles sont logarithmiques.

Une régression linéaire effectuée avec une transformation logarithmique des deux variables donne un coefficient de corrélation $R^2$=0.96 .

## 5. REPRESENTATION GRAPHIQUE DU RESEAU AERIEN

Un des aspects graphiques de PLANAIR a trait à la production de cartes géographiques du réseau aérien desservi. Comme nous le verrons au chapitre sept, ces cartes permettent de visualiser certaines caractéristiques du réseau. Cependant, la projection de la sphère terrestre sur une carte géographique plane amène plusieurs problèmes:

- Quelle projection choisir? Comment tenir compte de ses particularités?

- A partir du plus court chemin entre deux points sur la sphère (grand cercle), comment établir le plus court chemin sur le plan?

Sans aborder le détail des calculs mathématiques, nous allons exposer ici les approches utilisées pour contourner ces difficultés.

Il existe une multitude de projections géographiques différentes, et le problème du choix parmi celles-ci n'est pas simple car aucune ne peut être considérée comme supérieure aux autres ni ne peut s'appliquer dans toutes les situations. En navigation marine et aérienne, par exemple, la projection cylindrique de Mercator est très utilisée à cause de sa simplicité et de ses propriétés conformes (voir, par exemple, Adler et Richardus (1972); Robinson et Sale (1969)). Cependant, elle est inutilisable pour les régions polaires où elle entraîne trop de distortions. Plusieurs vols aériens internationaux devant passer dans la région boréale, une projection polaire serait donc aussi nécessaire. Et si l'on n'implante

que ces deux projections, il est fort possible qu'elles ne seront plus suffisantes pour une autre application que l'on voudra ajouter par la suite. La meilleure solution consiste alors à *enfermer* les calculs de projection dans une routine de bas niveau et à établir une interface avec des routines graphiques de plus haut niveau qui soit complètement indépendante du type de projection utilisée. On peut ainsi ajouter tous les types de projection que l'on veut en ne faisant que des modifications très locales. Cette solution n'est cependant pas aussi simple qu'il paraît; deux projections géographiques différentes mènent souvent à des formes graphiques particulières et très dissemblables l'une de l'autre. A tel point qu'il devient difficile de ne pas en tenir compte dans les routines graphiques de plus haut niveau. Reprenons l'exemple des projections polaire et de Mercator. Une projection conique polaire projette un hémisphère sur un cercle. Un plus court chemin entre deux points sur la carte peut toujours être tracé d'un trait continu (voir figure 5-1). Avec Mercator, par contre, le globe est projeté sur un rectangle où les pôles sont situés à l'infini. Sur une telle carte, il peut arriver que le plus court chemin entre deux points soit discontinu: il peut "disparaître" au bord de la carte pour revenir de l'autre côté ("wrap-around" en anglais) ou même, s'il passe par un pôle, se rendre à l'infini avant de revenir (voir figure 5-2). Les problèmes graphiques sont donc forts différents dans chaque cas. On peut néanmoins s'en tirer si on définit les routines de base avec précaution. Nous avons ainsi été amenés à définir trois routines qui nous permettent de traiter pratiquement tous les types de projections géographiques (voir figure 5-3).

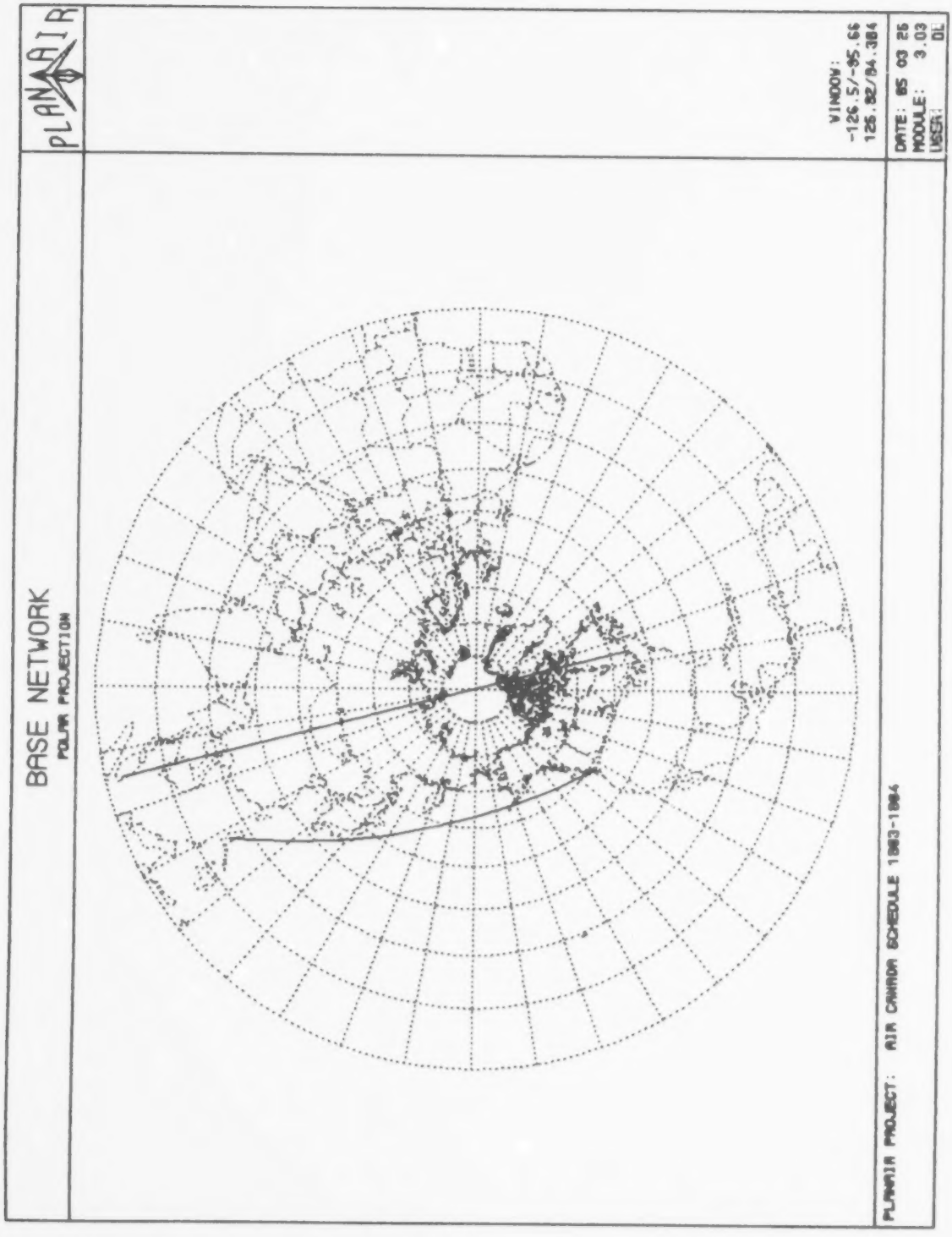


FIGURE 5-1: Projection conique polaire - Plus courts chemins de Montréal à Singapour et de Vancouver à Manille.

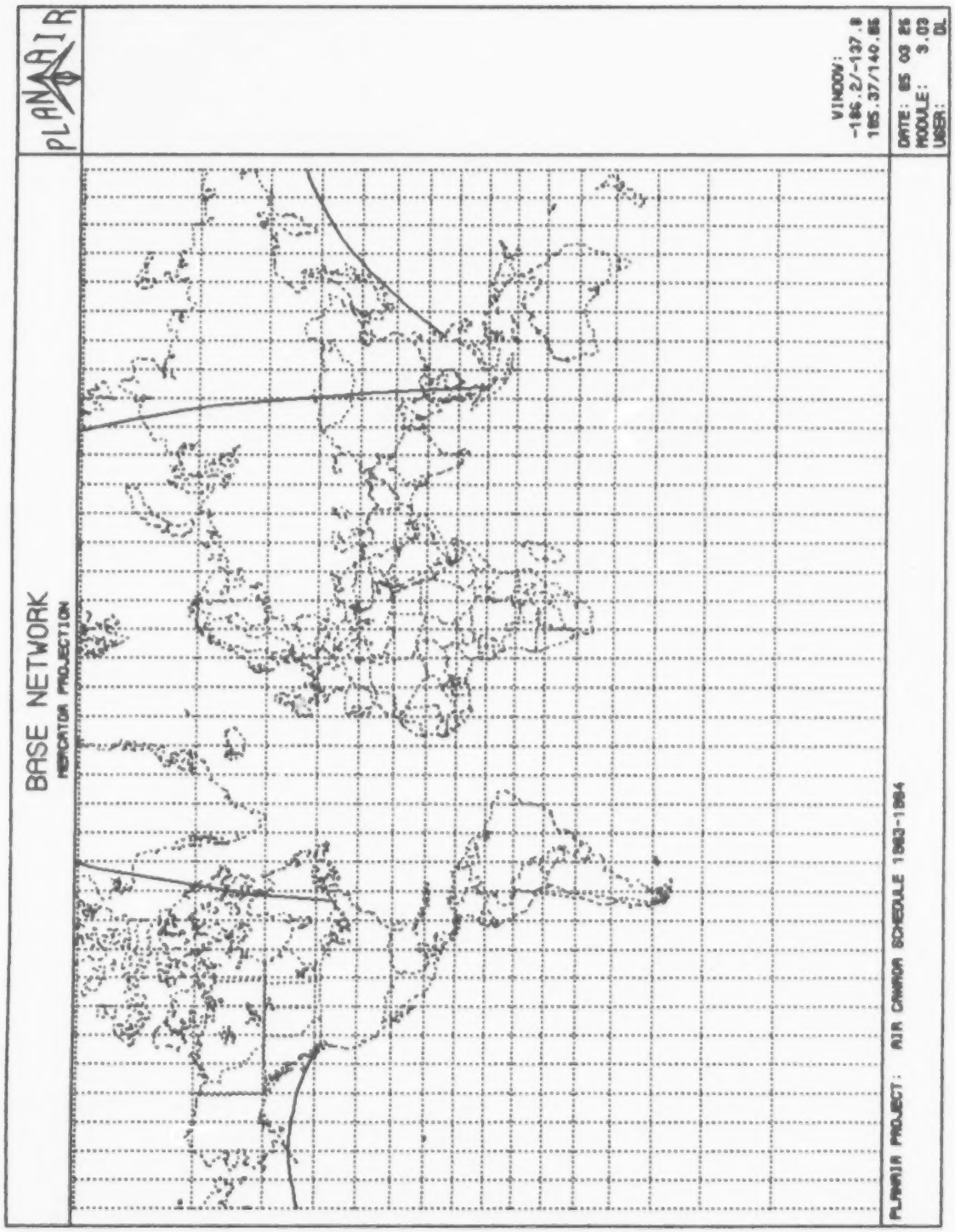


FIGURE 5-2: Projection de Mercator - Plus courts chemins de Montréal à Singapour et de Vancouver à Manille.

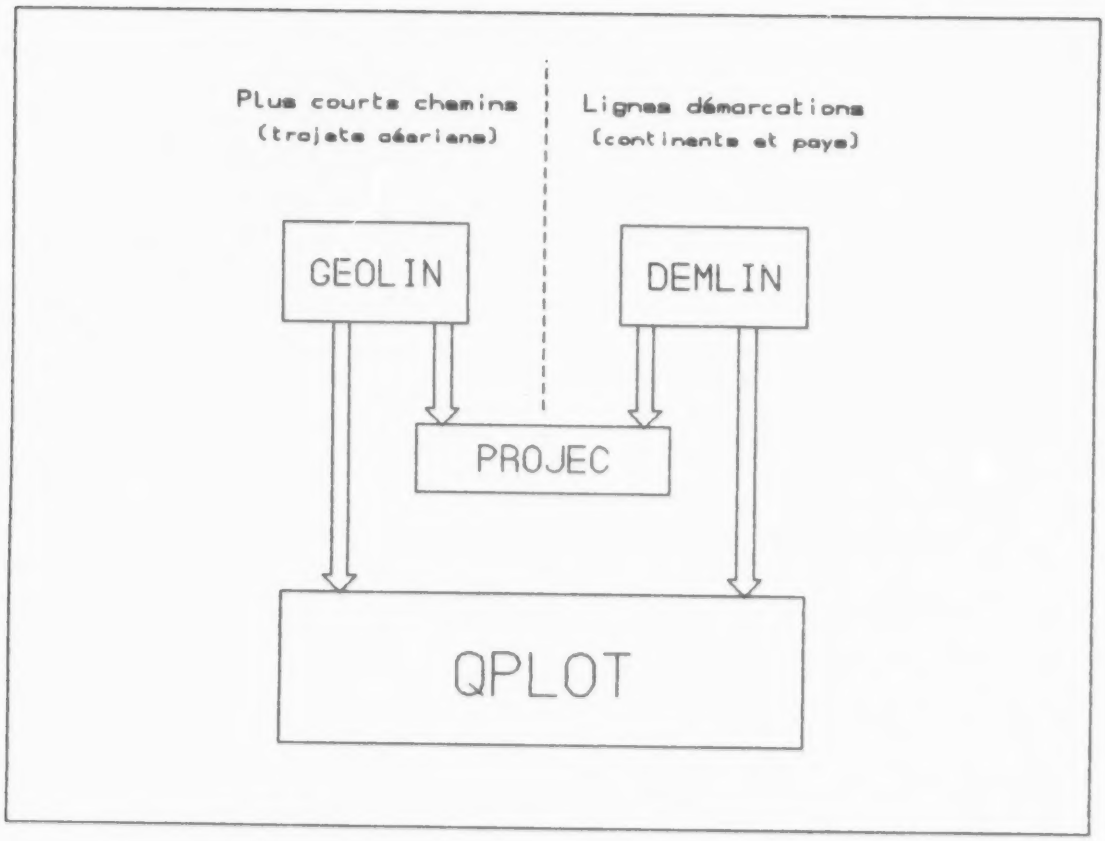


FIGURE 5-3: Hiérarchie des routines de projections géographiques.

La routine PROJEC est à la base de tout. C'est elle qui connaît les différents types de projections et leurs paticularités. Elle a trois principaux rôles:

- Transformation de coordonnees géographiques (latitude et longitude) à cartésiennes;
- Transformation de coordonnées cartésiennes à géographiques;
- Détermination des parties visibles d'un plus court chemin entre deux points sur la sphère ("clipping" en anglais).

C'est ce dernier point qui représente la plus grande partie du travail. Pour chaque plus court chemin entre deux points, PROJEC détermine le nombre de segments continus visibles qui le composent et les coordonnées des points terminaux de chacun. Pour la majorité des projections géographiques, les plus courts chemins seront composés d'au plus deux segments sur le plan. Il existe cependant certaines projections qui en nécessiteront plus, notamment les projections dites "interrompues" qui présentent des cartes prenant parfois la forme d'un "M" ou d'un "W" (voir, par exemple, Mailing,1973).

Lorsque deux points sont assez près sur la sphère terrestre, le plus court chemin qui relie leurs homonymes dans le plan cartésien peut être approximé par une droite, indépendamment de la projection utilisée. C'est cette particularité qui est utilisée par la routine DEMLIN qui trace les lignes de démarcations géographiques. Toutes les divisions géopolitiques du globe ont été codifiées à l'aide d'une table traçante avec environ 30 000 points. De ce nombre, 3000[1] ont été extraits et entrés dans la banque de données en coordonnées géographiques avec le module 1.41. Pour chaque paire de points consécutifs, DEMLIN appelle PROJEC pour effectuer la transformation en coordonnées cartésiennes et déterminer les segments visibles qui les relient. Puis elle appelle la routine graphique de base (QPLOT) pour les tracer avec une ligne droite.

Il reste encore à traiter le cas où les deux points à relier par un plus court chemin sont éloignés l'un de l'autre, pour tracer des trajets

1 Ce nombre de points utilisés convient pour la majorité des applications. Toutefois, pour les cas où une carte précise à petite échelle est nécessaire, il peut être augmenté.

aeriens, par exemple. Il n'est pas suffisant, en effet, de connaître les coordonnées cartésiennes de deux points et de savoir que le chemin qui les relie sur le plan est continu pour pouvoir le tracer. Il faut aussi connaître la courbure de ce chemin sur le plan car ce n'est pas nécessairement une ligne droite. C'est le rôle de la routine GEOLIN de déterminer cette courbure et de tracer le chemin. Elle procède par approximations successives: étant donné deux points A et B sur la sphère et leurs homonymes A' et B' dans le plan, on peut calculer, à l'aide de formules de geométrie des triangles spheriques, un point milieu C. Si, dans le plan, l'angle formé par les segments (A',C') et (C',B') est à un seuil près de $180^{o}$, on peut tracer ces deux segments à l'aide de deux droites. Dans le cas contraire, la courbure est encore trop grande pour être approximée par des droites, on applique le même traitement à chacun des segments en les divisant en deux. Ainsi de suite jusqu'à ce que les segments ainsi formés soient assez courts pour être approximés par des droites. On arrive ainsi, en appelant de façon répetitive la routine PROJEC, à tracer un chemin entre deux points geographiques, quelles que soient la distance qui les sépare et la projection utilisée. Le choix du seuil joue un rôle important: un seuil très petit permettra de tracer de belles courbes lisses, mais nécessitera un temps de calcul et d'affichage considérable. Nous avons fixé la valeur de ce seuil à $5^{o}$. Les courbes obtenues sont lisses, considérant la précision des équipements graphiques utilisés, tout en étant affichées en un temps raisonnable. Il s'agit toutefois d'un paramètre de module qui peut être ajusté pour des applications requérant plus ou moins de précision dans les graphiques.

Nous n'avons jusqu'ici programmé que deux types de projections: celle de Mercator et une projection stéréographique polaire conforme. Cela de-

vrait être suffisant pour la plupart des applications. Mais si une autre projection devenait nécessaire, il n'y aurait qu'à ajouter un peu de code à la routine PROJEC. Tout le reste est indépendant du type de projection utilisée.

## 6. INTERFACE ENTRE L'USAGER ET LE SYSTEME

Un grand soin a été apporté, lors de la conception du logiciel EMME/2, à l'élaboration d'un ensemble de mécanismes régissant la communication entre le système interactif-graphique et l'usager. Etant complètement indépendants du contexte des transports urbains, ces mécanismes ont pu être appliqués tels quels dans le cadre du projet PLANAIR. Nous avons ainsi pu mettre à profit toute l'expérience qui a été acquise pendant le développement de EMME/2 et réaliser des gains appréciables sur le temps d'implantation de PLANAIR.

Nous ne ferons ici qu'un court résumé des principales caractéristiques de l'interface entre le système et le monde extérieur. Nous référons le lecteur intéressé au chapitre trois de Spiess (1984) où le sujet est traité en profondeur.

Mentionnons d'abord quelques points importants concernant la structure du logiciel. PLANAIR est composé d'une vingtaine de programmes autonomes, ou *modules*, accédant à une même banque de données. Chacun est identifié par un numéro de la forme X.YY où X indique le groupe auquel le module appartient, et YY le distingue à l'intérieur de ce groupe. Nous expliquerons en détail les caractéristiques et le fonctionnement de chacun dans le cadre du chapitre suivant.

Du point de vue de l'usager, chaque module représente un outil, ou fonction, disponible dans le système pour exécuter une tâche précise. Un

seul module peut être exécuté à la fois, et chacun est appelé à partir du *menu principal* (qui est lui même un module, sans que ce ne soit apparent pour l'usager). Par défaut, dès qu'un module se termine, l'usager retourne au menu principal. Pour un usager plus expérimenté, toutefois, il est possible de prendre un "raccourci" en allant directement d'un module à un autre, sans passer par le menu principal.

Du point de vue informatique, chaque module est un programme autonome et aucune information n'est passée directement entre deux modules. Tout échange se fait par l'intermédiaire de la banque de données. PLANAIR est programmé en FORTRAN '77 et implanté sur un micro-ordinateur Pixel 100/AP. Il n'y a aucune dépendance liée à l'installation au niveau des modules; tous les aspects particuliers au système d'exploitation, à l'ordinateur ou à l'équipement graphique utilisé sont isolés dans un petit sous-ensemble de la bibliothèque des routines.

Abordons maintenant l'aspect communication entre le système et le monde extérieur. Au plus bas niveau sont définies les unités physiques où les programmes peuvent lire ou écrire. Parmi celles-ci on retrouve:

- l'entrée standard (clavier du terminal)
- la sortie standard (écran du terminal)
- la sortie graphique immédiate (écran du terminal graphique)
- l'entrée graphique immédiate
- l'imprimante
- le traceur de courbes
- le fichier séquentiel des données à l'entrée
- le fichier séquentiel des données à la sortie
- le fichier des diagnostics d'erreurs

Plusieurs des unités physiques mentionnées ci-haut correspondent à des équipements périphériques de l'ordinateur. Pour permettre l'utilisation de différents types de terminaux, de traceurs de courbes et d'imprimantes, une table des équipements est maintenue, qui décrit les caractéristiques de chacun des appareils périphériques auxquels on peut accéder pour une installation particulière.

A un plus haut niveau, toutes les entrées/sorties sont partitionnées en cinq catégories logiques auxquelles on associe:

- des caractéristiques de format
- des unités physiques qui peuvent être utilisées avec cette catégorie
- un protocole décrivant les règles que doit suivre tout échange d'information
- un ensemble de sous-routines pour implanter ce protocole

Voici une brève description de chacune des catégories:

DIALOGUE: Le dialogue interactif entre l'usager et le système consiste en une séquence de questions posées par le système, et auxquelles répond l'usager. La langue utilisée pour la communication entre l'usager et le système est l'anglais. Afin d'assurer une certaine uniformité, et de rendre le système plus "prévisible" pour l'usager, les questions posées ne peuvent prendre qu'une des trois formes bien définies suivantes:

a) La question **YES/NO**, utilisée quand l'usager se voit devant un choix "oui ou non", est caractérisée par un point

d'interrogation à sa fin. Seules les réponses Y(es), N(o), R(estart) pour reprendre à partir du début du module courant et Q(uit) pour retourner au menu principal sont admises.

b) La question **SELECT** donne à l'usager le choix parmi plusieurs options offertes par un menu. L'usager entre le numéro qui correspond à son choix, ou, s'il veut terminer le dialogue, R(estart) ou Q(uit).

c) La question **ENTER**, finalement, permet d'entrer une ou plusieurs valeurs numériques ou alphanumériques en format libre.

RAPPORT: Tout affichage tabulaire d'information alphanumérique entre dans cette catégorie. Chaque page de rapport porte une entête standard qui indique le module ayant généré celle-ci, la date et l'usager, ainsi que des descriptions pertinentes de projet. Un rapport peut être dirigé soit vers l'imprimante, soit vers le terminal.

GRAPHIQUE: Cette catégorie est utilisée pour toute entrée et sortie graphique. On distingue deux formats: *le dessin* ("plot" en anglais), destiné à toute sortie graphique qui a pour but d'afficher et de visualiser des éléments contenus dans la banque de données, et la *feuille de travail* ("graphic worksheet") pour toute modification interactive-graphique du contenu de la banque de données.

Tous les dessins sont faits à partir de vecteurs. Le minimum

requis d'un équipement graphique est qu'il sache tracer un vecteur reliant deux points sur l'écran. Il n'est pas nécessaire que l'équipement puisse générer des caractères alphanumériques, puisque le système graphique est muni d'un générateur de caractères par logiciel. Si l'équipement graphique permet des opérations plus complexes que le tracé d'un simple vecteur (génération de caractères, patrons de lignes, couleurs, etc.), ceci est indiqué dans la table des équipements, afin que le système graphique puisse en tirer avantage.
Un dessin peut être acheminé vers le terminal graphique ou vers le traceur de courbes.

TABLE: Cette catégorie permet de lire ou écrire des données en mode de traitement par lot ("batch" en anglais). Ce mode non-interactif est surtout utilisé pour l'entrée des données préparées à l'extérieur du système. C'est aussi un moyen de transférer des données entre deux banques de données différentes ou entre le logiciel et des programmes externes.

DIAGNOSTIC: Cette dernière catégorie n'est définie que pour la sortie. Normalement, quand l'usager commet une erreur, il reçoit un message d'erreur sous la forme d'une phrase qui lui en explique la cause. Un point d'exclamation ("!") sert à distinguer ce message d'erreur des autres messages. Dans la plupart des cas, celui-ci est suffisant pour permettre à l'usager de corriger l'erreur en question et continuer son travail. Afin de garder une trace de toutes les erreurs et pouvoir reconstruire les détails si cela devient nécessaire, un diagnostic est envoyé au

*fichier des diagnostics* chaque fois qu'un message d'erreur est envoyé à une autre catégorie d'entrée/sortie (dialogue, rapport ou graphique). Il contient le numéro du module, sa date de compilation, le mode d'opération (interactif ou lot), un numéro d'erreur et trois paramètres qui décrivent les circonstances précises entourant l'erreur.

Plusieurs auteurs cités par Shneiderman (1980) soulignent l'importance de pouvoir adapter le dialogue au degré d'expérience de l'usager avec le système. Deux mécanismes ont été implantés à cette fin: les paramètres de module et les commutateurs.

Les paramètres de module permettent à l'usager d'avoir un contrôle très fin sur les modalités d'application des opérations du module courant, tandis que le dialogue, lui, ne contient que les questions essentielles. Au début de l'exécution d'un module, ces paramètres sont initialisés à des valeurs par défaut. Un usager qui ne veut qu'utiliser ces valeurs par défaut n'a rien à faire. Si, par contre, l'usager désire changer les valeurs de ces paramètres, il peut faire cela en choisissant l'option correspondante dans la première question "SELECT" du module. Dans ce cas, il doit parcourir une séquence de dialogue qui lui permet de redéfinir tous les paramètres de ce module. Les nouvelles valeurs ainsi définies peuvent avoir un caractère temporaire (être oubliées en quittant le module), ou devenir les nouvelles valeurs de défaut.

Tandis que la sphère d'influence d'un paramètre de module est limité au module même, les commutateurs ("switches") agissent à un niveau plus global en permettant à l'usager "d'allumer" ou "d'éteindre" certaines

séquences de dialogue à travers tout le système. Par exemple, un usager qui est branché sur un terminal à basse vitesse perdrait beaucoup de temps chaque fois que le menu principal (la liste des modules disponibles) est affiché. En éteignant le commutateur correspondant, l'affichage du menu principal est supprimé, sauf si demandé explicitement.

## 7. LES MODULES

Comme mentionné plus tôt, PLANAIR est composé d'une vingtaine de modules organisés autour d'une même base de données. Dans cette section, nous donnerons une courte description de chacun de ces modules et présenterons quelques exemples de sorties graphiques et numériques. Notre but n'est pas de fournir ici un manuel d'usager; le système n'en étant qu'à ses premières phases de développement, beaucoup de modifications risquent encore d'être apportées. Nous voulons plutôt donner un aperçu du système PLANAIR en présentant les modules réalisés jusqu'ici et ceux en voie de développement. Le tableau 7-1 donne la liste des modules programmés et de ceux envisagés à court terme (ces derniers y sont marqués d'un astérisque). Comme le montre ce tableau, les modules sont partitionnés en cinq groupes, chacun touchant à un aspect précis des données du système. Le groupe 1 englobe les utilitaires qui ont trait à la gestion des fichiers internes de la banque de données et de l'environnement de travail. Le groupe 2 concerne les données se rapportant aux noeuds desservis et aux véhicules de la flotte. Les groupes 3 et 4 comprennent tous les outils nécessaires à la mise en place, à la gestion et à la consultation des données relatives aux activités et aux accrochages. Le groupe 5, enfin, comporte les modules permettant de considérer un horaire du point de vue des appareils de la flotte.

| | |
|---|---|
| 1. | LES UTILITAIRES |
| 1.11 | Manipulations directes de la banque de données |
| 1.21 | Consultation du journal de bord |
| ‡1.23 | Titres et unités |
| 1.31 | Initialisation |
| 1.41 | Entrée / modification / affichage des démarcations |
| 2. | LES DONNEES DE BASE |
| 2.01 | Entrée / modification / sortie des classes de véhicules |
| 2.02 | Entrée / modification / sortie des noeuds |
| 3. | LES ACTIVITES |
| 3.01 | Entrée séquentielle des activités |
| ‡3.02 | Entrée / modification interactive des activités |
| 3.03 | Affichage des activités |
| 3.04 | Sortie alphanumérique des activités |
| 4. | LES ACCROCHAGES |
| 4.01 | Entrée séquentielle des accrochages |
| ‡4.02 | Entrée / modification interactive des accrochages |
| 4.03 | Affichage des accrochages |
| 4.04 | Sortie alphanumérique des accrochages |
| ‡4.05 | Modification interactive - graphique des accrochages |
| 4.06 | Génération automatique des accrochages |
| 5. | LES TACHES DES VEHICULES |
| ‡5.03 | Sortie alphanumérique des tâches des véhicules |
| ‡5.04 | Affichage des tâches des véhicules |
| ‡5.05 | Modification interactive - graphique des tâches des véhicules |

TABLEAU 7-1: Les modules de PLANAIR.

## 7.1 Les utilitaires

Les modules du groupe 1 sont liés à la gestion de l'environnement de travail et sont relativement indépendants du problème de planification. De ce fait, la plupart ont pu être adaptés de EMME/2 avec peu de modifications.

Le module 1.11 "Manipulations directes de la banque de données" permet d'accéder directement aux fichiers internes de la banque de données et aux données brutes qui y sont contenues. C'est un outil de mise au point qui facilite la détection d'erreurs et l'application de mesures correctives de sauvetage, lorsque cela est possible. A ce titre, c'est un module qui ne devrait pas servir lors d'une séance d'utilisation normale. Entre des mains inexpertes, il peut rendre une banque de données complètement inutilisable.

Le module 1.21 "Consultation du journal de bord" donne accès à l'histoire récente de la banque de données. Une trace des appels de modules effectués par les différents usagers est automatiquement conservée dans un journal de bord. Les 300 à 500 derniers appels (dépendant de la longueur des commentaires associés aux entrées) sont conservés. On peut, à tout moment, consulter cette information en permettant l'inclusion ou l'exclusion de plusieurs critères: les initiales des usagers, les dates ou les numéros de modules. Les résultats de cette sélection apparaissent par ordre chronologique et peuvent être visualisés à l'écran ou envoyés à l'imprimante.

Le module 1.23 "Titres et unités" n'est pas encore programmé, mais

tout est prévu pour sa mise en place prochaine. Il permettra à un usager d'établir l'unité de distance avec laquelle il désire travailler et les titres principaux qui apparaîtront sur les rapports et les graphiques. De plus, il sera possible de redéfinir des termes tels que *noeud, véhicule, activité et accrochage* qui apparaissent dans les dialogues pour les remplacer par des termes avec lesquels l'usager est plus familier. Cette particularité permettra d'adapter le système à chaque application, et non l'inverse, ce qui contribuera à une plus grande acceptation de celui-ci.

Le module 1.31 "Initialisation" a pour but de fixer des paramètres globaux qui s'appliquent à l'ensemble des données de la banque. C'est là que sont définies, par exemple, les dates définissant la période couverte par l'horaire des vols contenus dans la banque. C'est aussi à l'aide de ce module que l'usager peut attribuer des valeurs au vecteur de priorité des jours dont nous avons discuté dans la section sur le décodage des vecteurs de bits.

Le module 1.41 "Entrée / Modification / Affichage des lignes de démarcation" est utilisé pour entrer les caractéristiques géopolitiques qui seront affichées sur certains graphiques. Ces démarcations servent, par exemple, à tracer les contours des continents et des pays sur les cartes géographiques produites par le module 3.03 "Affichage des activités". L'usager dispose ainsi de neuf ensembles de démarcations qu'il peut superposer aux graphiques afin d'y ajouter de l'information. Actuellement, les opérations d'affichage et de modification interactive-graphique ne sont pas encore implantées; seule l'opération d'entrée par lots a été programmée.

## 7.2 Les données de base

Le groupe 2 comprend deux modules permettant de gérer l'information concernant les véhicules et les noeuds. Ces deux ensembles de données constituent la partie la plus statique de la banque de données, c'est-à-dire la partie des données qui sera la plus stable et la moins souvent modifiée. Ce groupe de modules sert donc surtout au cours de la création de la banque de donnees.

Le module 2.01 "Entrée / Modification / Sortie des classes de véhicules" sert à entrer les données qui se rattachent aux classes de véhicules, que ce soit par lots ou de façon interactive. Il permet aussi de modifier ces mêmes données par dialogue interactif et de les consulter en produisant l'information à l'écran, à l'imprimante ou dans un fichier sous forme d'images cartes.

Le module 2.02 "Entrée / Modification / Sortie des noeuds" est le pendant du module précédent pour les noeuds. Les options offertes à l'usager et son fonctionnement sont identiques.

## 7.3 Les activités

Le groupe 3 réunit tous les outils nécessaires à la gestion des informations liées aux activités.

Le module 3.01 "Entrée séquentielle des activités" lit un fichier séquentiel contenant la description des activités et de leurs attributs et les incorpore à la table des activités. Le rapport produit contient une copie des images cartes lues en entrée, d'éventuels messages d'erreur aux endroits appropriés et un sommaire indiquant le nombre d'images cartes lues et le nombre d'erreurs trouvées. Pour un horaire tel que celui d'Air Canada qui comporte, sur une période d'un an, plus de 2000 activités (soit au-delà de 314 000 événements), l'opération requiert environ dix minutes (temps réel) sur un micro-ordinateur Pixel 100/AP.

Le module 3.02 "Entrée / Modification interactive des activités" n'est pas encore implanté. Il permettra de modifier numériquement le contenu de la table des activités.

Le module 3.03 "Affichage des activités" produit des graphiques du réseau aérien couvert sous forme de cartes géographiques en faisant appel aux routines dont il a été question au chapitre cinq. Ainsi, n'importe quel sous-ensemble des activités peut actuellement être tracé sur une carte avec les démarcations géopolitiques (voir figure 7-1).

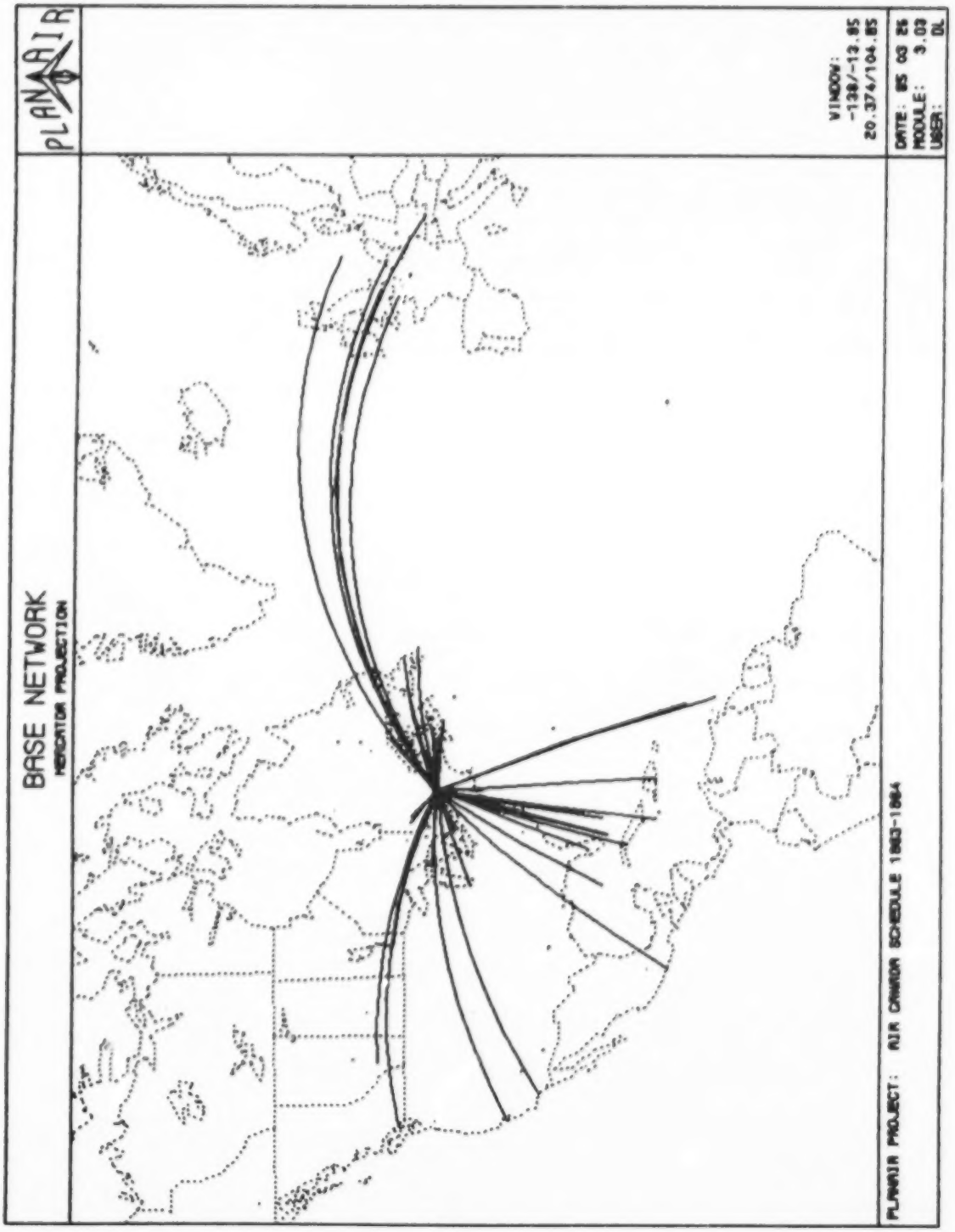


FIGURE 7-1: Réseau d'accès direct aux aéroports de Dorval et Mirabel.

Ce n'est là cependant qu'un des aspects de ce module qui n'est que partiellement développé. Nous prévoyons trois principales sphères d'application pour ce module:

- Le dessin du réseau dans l'espace est un moyen privilégié pour afficher de l'information concernant les différentes paires origine-destination. Cela permet de visualiser des caractéristiques du réseau, telles que son étendue ou sa capacité par exemple, et de les afficher de façon condensée. On peut même envisager, pour une planification à plus long terme, d'entrer certaines données des réseaux des sociétés concurrentes de façon à obtenir une vue globale de la situation et pouvoir établir de meilleures stratégies de marketing.

- Du fait des grandes qualités visuelles d'une telle représentation, ce module peut devenir un outil efficace pour vérifier la cohérence des horaires contenus dans la banque de données lors des stades initiaux de sa création.

- Toute l'information concernant le cheminement de chaque appareil de la flotte est contenue dans la banque de données. Il est donc très facile d'afficher sur une carte géographique la position de chacun à un temps donné. Cette information peut être très utile sur une base opérationnelle dans le cas, par exemple, ou un appareil doit être remplacé par un autre dans un délai très bref.

Le module 3.04 "Sortie alphanumérique des activités" permet d'obtenir des rapports qui sont produits à l'écran ou à l'imprimante. Plusieurs formats sont disponibles:

- Par ordre alphabétique des identificateurs des activités, chacune avec tous ses attributs;

- Par ordre alphabétique des identificateurs des activités, chacune avec les dates des événements libres et de ceux qui sont accrochés;

- Par ordre alphabétique des identificateurs des noeuds, dans une forme semblable à celle des horaires distribués au public.

Un exemple d'un de ces rapports est donné au tableau 7-2.

PLANAIR Module: 3.04 Date: 85 03 26 User: DL Page: 117
Project: Air Canada schedule 1983-1984

Activities listed by nodes

| NODE | from | to | dep. | arr. | activity | -----date----- | vehicle |
|---|---|---|---|---|---|---|---|
| aca | | | | | | | |
| | | yyz | 1500 | 2020 | 7051 | DE22 DE22 4 | 7 |
| | | yyz | 1500 | 2020 | c051 | DE29 FE09 4 | 7 |
| | | yyz | 1610 | 2130 | 7055 | DE17 DE17 6 | 7 |
| | | yyz | 1620 | 2140 | 7055 | DE18 DE18 7 | 23 |
| | | yyz | 1620 | 2140 | c055 | DE25 FE05 7 | 23 |
| | | yqb | 1630 | 2235 | c051 | FE16 MR29 4 | 7 |
| | | yyz | 1640 | 2200 | c055 | DE23 JA13 5 | 7 |
| | | yyz | 1730 | 2250 | 7055 | DE16 DE16 5 | 7 |
| | | yax | 1730 | 2320 | c055 | FE11 MR24 6 | 7 |
| anu | | | | | | | |
| | | bgi | 1355 | 1450 | 960 | MA05 JN09 6 | 16 |
| | | bgi | 1355 | 1450 | 960 | SE15 OC27 6 | 4 |
| | | yyz | 1630 | 2110 | 961 | JN16 SE08 6 | 16 |
| | | yyz | 1730 | 2125 | 961 | NO05 DE17 6 | 16 |
| | | | | | | DE18 AL28 67 | |
| bda | | | | | | | |
| | | yhz | 1135 | 1340 | 959 | FE18 MR30 6 | 16 |
| | | | | | | JN02 JN30 6 | |
| | | yhz | 1140 | 1345 | 959 | MR31 AL28 6 | 8 |
| | | yyz | 1425 | 1615 | 947 | MA01 JN01 2356 | 18 |
| | | | | | | JN02 JN17 235 | |
| | | | | | | JN18 JL06 1235 | |
| | | | | | | JL07 SE09 147 | |
| | | yyz | 1425 | 1625 | 947 | NO02 FE19 3 | 18 |
| | | | | | | FE20 AL23 1 | |
| | | yyz | 1435 | 1625 | 947 | AL29 JN17 17 | 4 |
| | | | | | | JN18 SE09 7 | |
| | | | | | | SE10 OC27 17 | |
| | | yhz | 1600 | 1805 | 959 | MA05 MA26 6 | 9 |
| | | yyz | 1755 | 1945 | 947 | JN02 JN30 6 | 18 |
| | | yyz | 1755 | 1955 | 947 | FE18 MR24 6 | 18 |
| | | yyz | 1815 | 2005 | 947 | MR31 AL28 6 | 8 |
| bgi | | | | | | | |
| | | yyz | 1525 | 2050 | 967 | JN22 SE07 5 | 4 |
| | | yyz | 1525 | 2045 | 967 | MA04 JN15 5 | 9 |
| | | yyz | 1620 | 2050 | 967 | NO05 DE31 6 | 8 |
| | | | | | | AD DE23 | |
| | | | | | | AD DE30 | |
| | | yax | 1635 | 2150 | 975 | JN17 SE09 7 | 16 |

TABLEAU 7-2: Exemple d'un rapport d'activités.

## 7.4 Les accrochages

Tous les outils nécessaires à la gestion des informations liées aux accrochages se retrouvent dans le groupe 4.

Le module 4.01 "Entrée séquentielle des accrochages" lit un fichier séquentiel qui contient la définition des accrochages à créer. Dans ce fichier, les accrochages peuvent être définis de façon précise pour les créer spécifiquement, ou de manière plus globale en donnant deux groupes d'activités à relier. Dans ce dernier cas, la matrice définie par ces deux groupes d'activités est parcourue colonne par colonne pour créer des accrochages entre toutes les paires d'activités compatibles.

Le module 4.02 "Entrée / Modification interactive des accrochages" n'est pas encore implanté. Il permettra, en suivant un dialogue, de modifier numériquement le contenu de la table des accrochages.

Le module 4.03 "Affichage des accrochages" produit des diagrammes espace-temps des activités et des accrochages. Deux exemples de tels diagrammes sont donnés aux figures 7-2 et 7-3. L'espace y est représenté sur l'axe vertical par une série de noeuds, tandis que l'axe horizontal représente l'échelle du temps. Chaque trait diagonal représente un événement entre les deux noeuds qu'il joint. Chaque trait horizontal plein établit une connexion entre deux événements. Il est ainsi possible de visualiser les événements qui sont accrochés ou non, et ce, pour n'importe quel sous-ensemble d'activités.

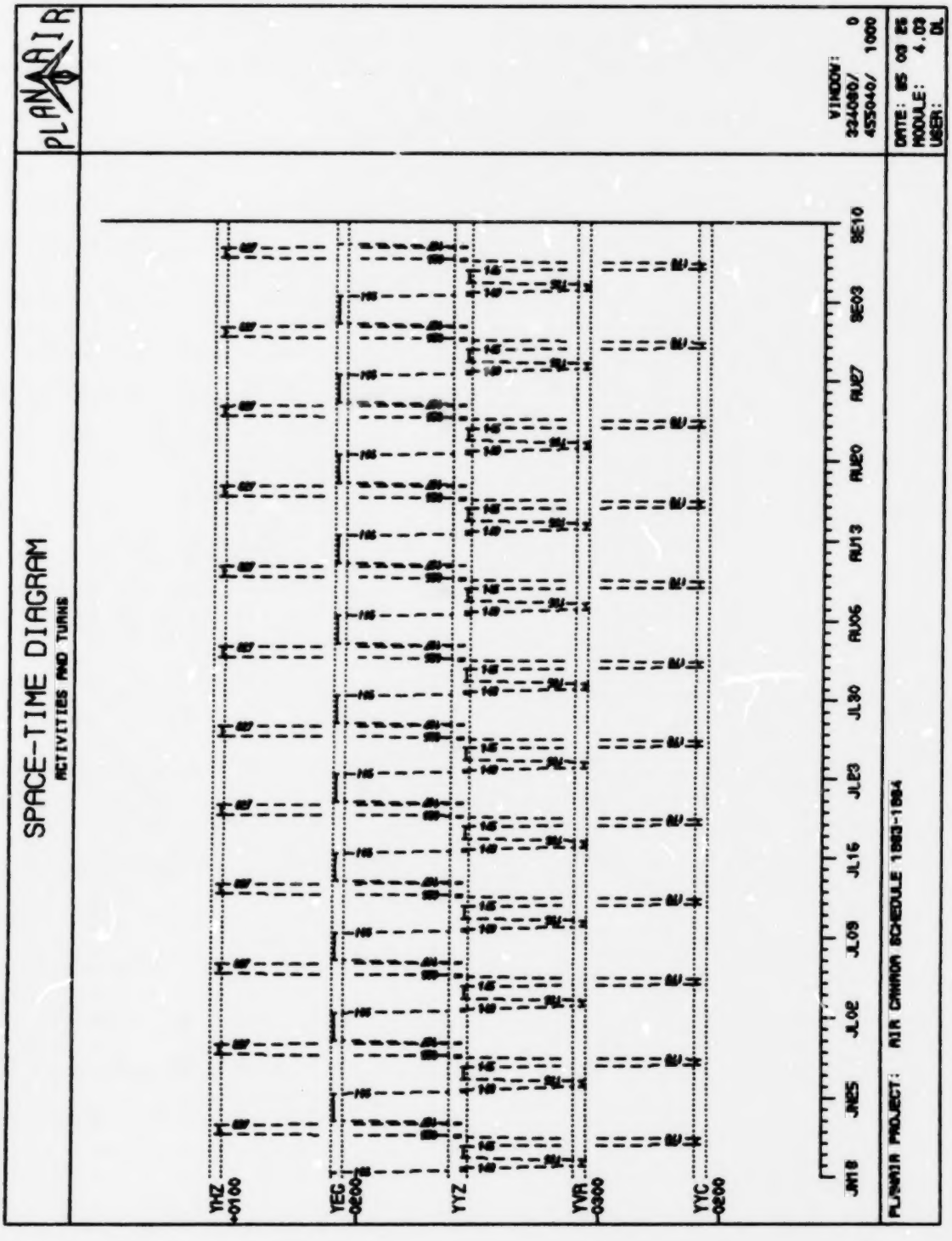


FIGURE 7-2: Diagramme espace-temps - Allocation d'un appareil LOCKHEED-1011

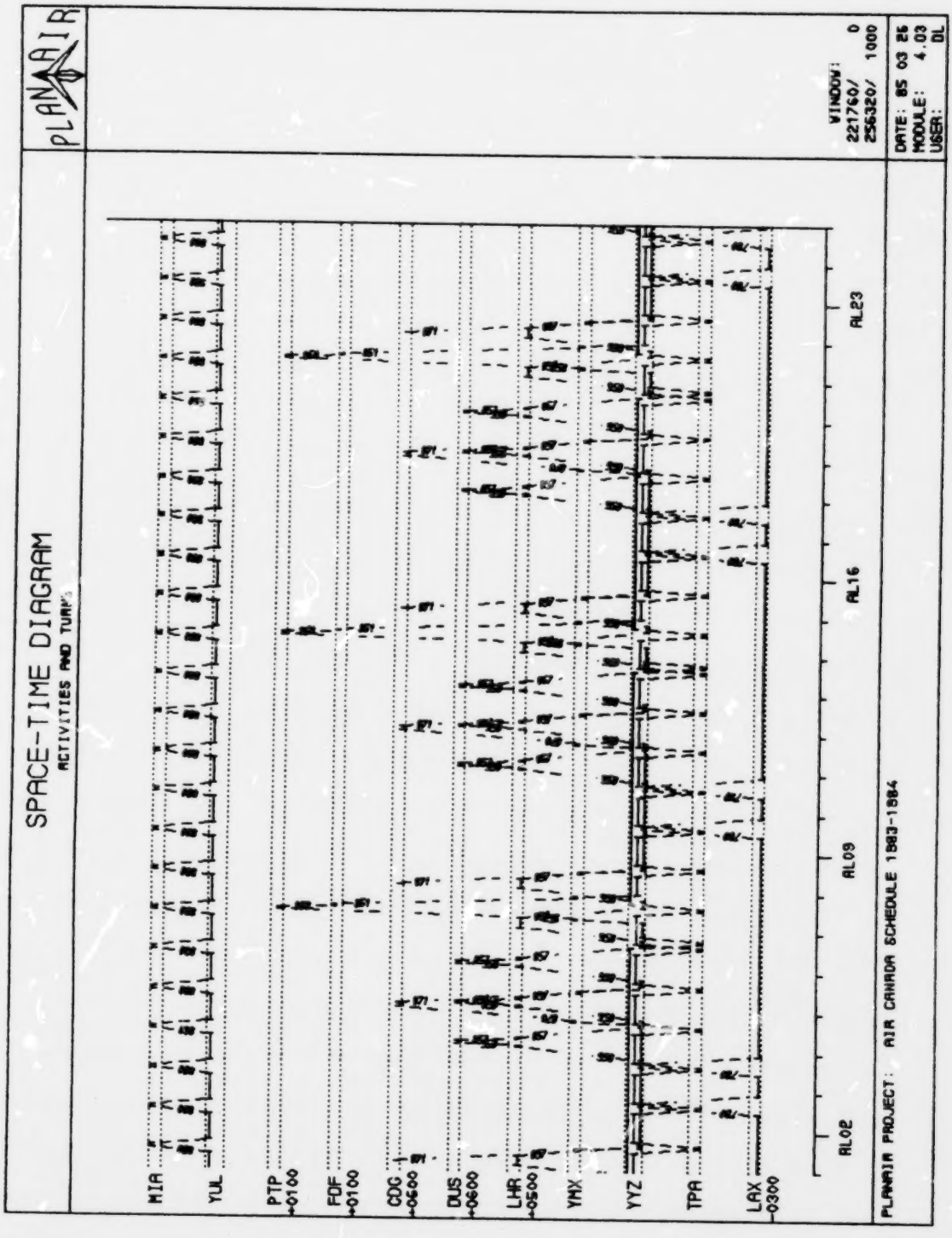


FIGURE 7-3: Diagramme espace-temps - Allocation de la flotte des BOEING-747

Le développement de ce module en est encore à ses débuts; beaucoup d'options sont encore à implanter pour utiliser tout le potentiel de ce type de diagramme. Comme c'est avec ce diagramme qu'un planificateur sera le plus souvent appelé à travailler, nous avons conçu ce module avec l'intention de fournir un ensemble d'options qui permettront à l'usager d'avoir un très grand contrôle sur l'information qui est affichée et sa disposition. Ce doit être un outil flexible qui s'adapte à une gamme variée d'utilisations. Nous discuterons ici des options qui ont été développées jusqu'à présent et de celles qui sont prévues à court terme.

- Axe d'espace

  L'usager définit l'axe vertical en spécifiant un ensemble ordonné de noeuds. Les noeuds apparaissent sur le graphique de bas en haut, dans l'ordre où ils ont été donnés. On peut ainsi contrôler le nombre de noeuds affichés et leur position relative. Bien que la matrice des distances séparant les noeuds soit disponible dans la banque de données, ceux-ci sont actuellement disposés sur le diagramme à des intervalles équidistants, sans tenir compte de l'éloignement réel. Nous prévoyons offrir la possibilité à l'usager d'obtenir un diagramme qui soit plus complet et qui tienne compte de cette information, en variant l'éloignement relatif des noeuds sur l'axe vertical. Nous voulons aussi permettre une plus grande flexibilité au niveau de l'affichage de l'information entre différentes paires de noeuds. Il sera, par exemple, possible de *déplier* un diagramme dans l'espace afin de pouvoir représenter toutes les activités dans la même direction et ainsi obtenir un dessin qui soit plus clair dans certaines circonstances. La figure 7-4 illustre cette idée.

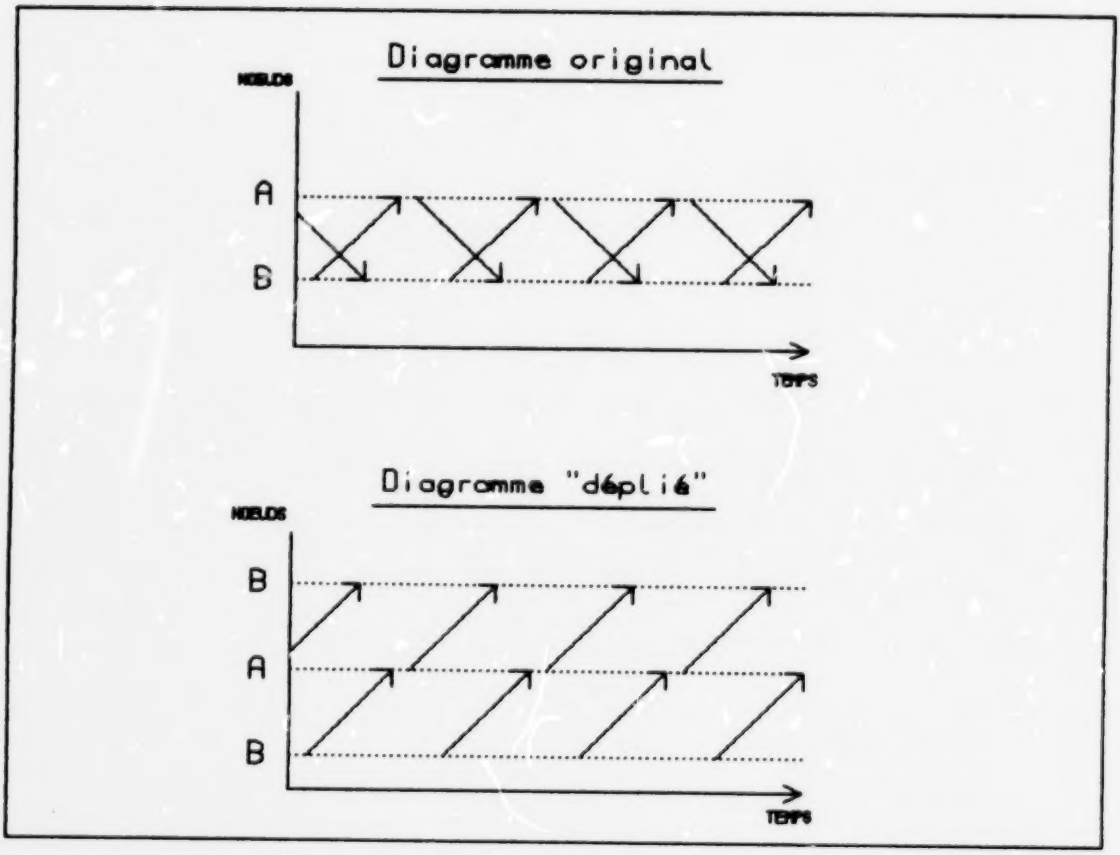


FIGURE 7-4: Diagramme espace-temps "déplié".

- Axe de temps

La fenêtre de temps à afficher est définie en spécifiant deux dates formées d'un mois, d'un jour et, facultativement, d'une heure dans la journée. Les unités de l'échelle de temps s'ajustent automatiquement avec une précision qui peut aller de la minute au mois.

Pour une planification à long terme, la notion de *période type* est importante. Aussi, pour mettre en évidence ces périodes types, il sera possible de superposer plusieurs parties du diagramme dans le temps. Par exemple, plutôt que de représenter les quatre semaines

d'un mois bout à bout selon une échelle continue, on pourra les superposer sur une seule semaine et ainsi mettre en évidence les ressemblances et les différences qu'elles comportent au niveau de l'horaire.

- Activités affichées

Le sous-ensemble d'activités à afficher est choisi à l'aide de critères tels que: noeud de départ ou d'arrivée, identificateur, heure de départ ou d'arrivée, ou une combinaison logique de ces derniers. Cette méthode s'est avérée très efficace pour désigner rapidement presque n'importe quel sous-ensemble d'activités. Nous la raffinerons encore un peu en élargissant le choix des critères et en permettant de prédéfinir des sous-ensembles qui pourront ensuite être rappelés simplement en les nommant.

- Informations alphanumériques affichées

Présentement, l'usager a le choix d'afficher ou non des symboles qui indiquent le lieu de provenance ou de destination des activités, et les identificateurs de celles-ci. A cela s'ajoutera la possibilité d'afficher d'autres informations alphanumériques choisies parmi les attributs des activités et des accrochages.

- Mode requêtes

Nous travaillons actuellement au développement d'un mode graphique d'interrogation. Ce mode requêtes permettra à l'usager de pointer une activité ou un accrochage affiché à l'écran et d'obtenir, dans la marge du dessin, les valeurs des attributs qui y sont reliés.

Il reste beaucoup à développer pour faire de ce module un outil fonc-

tionnel. Cependant, la partie développée jusqu'à maintenant nous a permis d'exploiter de nombreuses facettes des structures de données utilisées et d'en apprécier la justesse et la valeur.

Le module **4.04** "Sortie alphanumérique des accrochages" permet de produire des rapports concernant la table des accrochages. Comme nous l'avons mentionné au chapitre quatre, les accrochages n'ont pas d'identificateur propre. On choisit les accrochages à inclure dans le rapport en spécifiant les identificateurs des activités accrochées.

Le module **4.05** "Modification interactive - graphique des accrochages" n'est pas encore programmé. Il offrira le même graphisme et les mêmes options que le module 4.03, sauf qu'il se présentera sous la forme d'une *feuille de travail* et permettra d'effectuer des modifications graphiques qui seront effectuées sur le contenu de la banque de données. Les opérations permises seront les suivantes:

- Définition d'une *fenêtre d'action*

  La *fenêtre d'action* détermine la portée qu'auront toutes les commandes de corrections subséquentes, jusqu'à ce qu'une nouvelle fenêtre d'action soit définie. L'usager peut déclarer cette fenêtre comme étant locale, s'il ne désire agir qu'au niveau d'un événement ou d'une connexion, globale, si les corrections doivent s'appliquer à tous les événements (ou connexions) composant une activité (ou un accrochage), ou tout autre intermédiaire. Prenons, par exemple, une activité qui a lieu tous les jours de la semaine de janvier à juin et supposons que le planificateur veuille apporter une correction aux événements de celle-ci qui surviennent les jours de fin de semaine au mois de mai.

Il n'a alors qu'à définir sa fenêtre d'action (en entrant "DEFINE MA1 MA31 67" supposons), pointer un événement quelconque de l'activité visée avec le curseur graphique et effectuer la correction. Celle-ci est alors effectuée sur tous les événements de l'activité coincidant avec la fenêtre d'action et ce, même si certains d'entre eux ne sont pas directement visibles sur l'écran. Cette façon de procéder permet d'offrir à l'usager un ensemble restreint de commandes puissantes plutôt qu'un grand nombre de commandes différentes plus difficiles à retenir.

- Eliminer un accrochage ou une connexion

  Il suffit de pointer l'élément indésirable sur l'écran avec le curseur graphique pour que celui-ci soit immédiatement rayé de la banque de données.

- Créer un accrochage ou une connexion

  En pointant les deux activités (ou événements) à relier, l'opération est effectuée après avoir vérifié qu'il n'y a aucune incompatibilité de temps ou de type de véhicule entre les deux.
  On peut aussi ne pointer qu'une activité (ou un événement) à relier. Le système génèrera alors automatiquement un (ou des) accrochage(s) avec les événements libres les plus près.

- Déplacer le temps d'arrivée et de départ d'une activité ou d'un événement

  Si l'élément pointé n'est pas accroché, l'opération s'effectue sans problème. Par contre, s'il est accroché quelque part, l'opération n'est effectuée que si le déplacement n'entraîne aucun conflit à une

des connexions. En cas de conflit, l'usager est averti du fait et la connexion en cause est identifiée.

Il faut noter que, si l'usager ne déplace ainsi qu'un sous-ensemble d'événements d'une activité, le système scinde automatiquement l'activité en deux activités ayant le même identificateur, mais n'ayant pas les mêmes heures de départ et d'arrivée. Ceci est nécessaire puisque les événements déplacés, ne partageant plus la même heure de départ et d'arrivée que les autres, ne peuvent plus faire partie de l'activité.

- Consulter ou modifier les attributs d'une activité ou d'un accrochage
  Les attributs de l'élément pointé sont affichés dans la marge du dessin et de nouvelles valeurs peuvent être entrées au clavier.

- Obtenir une liste de candidats pour un accrochage
  En pointant une activité qui n'est pas saturée, un candidat compatible est donné dans la marge pour chaque événement non accroché faisant partie de la fenêtre d'action. On peut alors créer le (ou les) accrochage(s) suggéré(s), ou demander un autre candidat.

Plusieurs des fonctions graphiques nécessaires à l'implantation de ce module sont déjà disponibles dans la bibliothèque de sous-routines de EMME/2. Le travail de programmation est, de ce fait, grandement allégé, ce qui accélérera la phase de développement.

Le module 4.06 "Génération des accrochages" utilise l'algorithme décrit à la section 4.3 pour créer automatiquement une série d'accrochages. L'usager peut appliquer sélectivement cet outil analytique à n'importe quel

sous-ensemble d'activités. De cette façon, il est possible de s'en servir sur la totalité des activités pour générer une solution initiale, ou localement, sur un sous-problème donné.

Sous peu, d'autres algorithmes d'allocation viendront s'ajouter à l'algorithme LIFO déjà implanté et étendre la gamme des possibilités offertes à l'usager.

## 7.5 Les tâches des véhicules

Les modules du groupe 5 visent à apporter un complément d'information. Ils n'ont pas pour but de gérer de nouvelles données comme les groupes précédents, mais plutôt de fournir un moyen d'aborder différemment les mêmes données. Utilisés seulement une fois que plusieurs accrochages ont été établis, ils traitent l'information du point de vue des véhicules en permettant d'obtenir la liste de tâches (événements et connexions) auxquelles chacun est affecté.
Nous n'énoncerons ici que les grandes lignes des modules de ce groupe qui n'est pas encore développé.

Le module 5.03 "Sortie des tâches des véhicules", en plus de donner la liste des tâches auxquelles chaque appareil est affecté, fournira une série de statistiques telles que le nombre d'heures de vol maximum et moyen entre les périodes d'entretien et le nombre d'heures de vol au total.

Le module 5.04 "Affichage des tâches des véhicules" produira des graphiques appareils-temps tels que celui illustré à la figure 7-5. Ce type de diagramme permet de mieux visualiser les tâches que doit effectuer un appareil que le diagramme espace-temps. C'est la même information, mais

projetée selon un autre axe.

Le module **5.05** "Modification interactive - graphique des tâches des véhicules" sera basé sur le même principe que le module 4.05, mais appliqué à un diagramme appareils-temps.

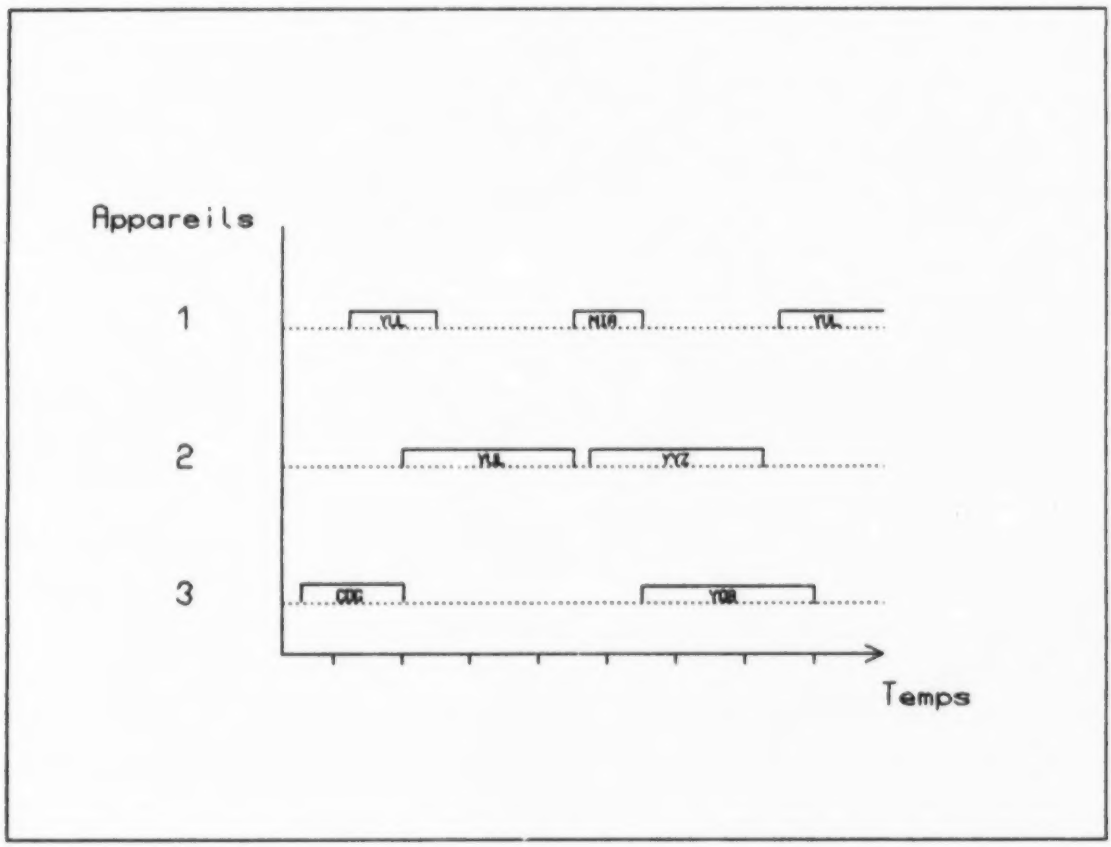


FIGURE 7-5: Diagramme appareils-temps prevu.

## 8. CONCLUSION

Nous avons développé, pour le problème de l'allocation des avions à un horaire prévu, un modèle permettant d'exploiter la nature quasi-périodique de l'horaire des vols. Contrairement à toutes les méthodes de planification à court terme portées à notre attention, ce modèle évite de devoir traiter les cas d'exception à l'horaire de façon explicite. Nous avons défini les termes *activité* et *événement* pour désigner l'horaire d'un vol et ses réalisations ponctuelles, et montré comment tous les événements d'une activité peuvent être traités en parallèle à l'aide d'opérations logiques. Cette particularité, combinée avec les propriétés de la stratégie LIFO, nous a permis de concevoir un algorithme d'allocation rapide et efficace.

Sur la base de ce modèle, nous avons développé le logiciel PLANAIR qui permet une optimisation interactive du problème d'allocation, en mettant à la disposition du planificateur un ensemble d'outils analytiques et interactifs-graphiques. PLANAIR comprend actuellement un ensemble de modules qui permettent de créer, modifier et consulter une banque de données composée de l'horaire des vols d'une société aérienne sur une période de planification pouvant aller jusqu'à un an. A cela s'ajoutent quelques modules graphiques et un module d'allocation automatique. PLANAIR n'en est toutefois qu'aux premiers stades de son implantation; il reste beaucoup à faire encore pour qu'il réalise son plein potentiel. D'une part, l'ajout des modules de modifications graphiques permettra d'obtenir un système plus dynamique et plus souple. D'autre part, il est aussi prévu

de compléter le système avec des outils d'optimisation qui fourniront au planificateur une vue plus globale du problème. Dans ce domaine, les bases théoriques établies par notre modèle pourront constituer un point de départ.

Deux facteurs ont joué un rôle important dans le développement de ce projet et sont à souligner:

- Nous avons eu, dès le début du projet, des contacts avec des personnes impliquées dans le processus de planification à l'intérieur d'une grande société aérienne. Tout au long du développement du modèle et du logiciel, nous avons ainsi pu nous assurer qu'aucun fossé ne venait séparer la théorie de la pratique, et pu prendre des mesures correctives dès qu'une telle tendance se dessinait. De même, les données utilisées pour effectuer les tests proviennent d'une situation réelle. Plutôt que de travailler sur un problème fictif et d'extrapoler ensuite à une situation réelle, nous avons préféré utiliser dès le départ des données complètes. De cette façon, il a été possible de corriger rapidement des inconsistances qui ne seraient sans doute pas apparues avec des données fictives ou incomplètes.

- D'emblée, nous avons pu bénéficier de l'expérience qui avait été acquise pour la conception du logiciel EMME/2 et avons eu accès à la bibliothèque des routines. Les concepts élaborés pour EMME/2 ont donc eu une grande influence sur PLANAIR dont l'implantation a été grandement accélérée grâce aux routines déjà existantes.

A plus long terme, on peut considérer que PLANAIR ne constitue qu'une étape vers un décloisonnement du processus de planification des opérations d'une grande société aérienne. Le problème est, en effet, tellement grand que chaque département de la compagnie travaille actuellement de façon plus ou moins isolée sur la partie le concernant directement. On a ainsi plusieurs sous-problèmes traités de façon presqu'indépendante, alors qu'ils devraient être intégrés dans le cadre d'un tout pour arriver à une solution globalement optimale. On entrevoit donc la possibilité d'ajouter d'autres modules à PLANAIR qui pourra alors devenir un outil réunissant les planificateurs des différents départements autour d'un même système. Bien sûr, il s'agit là d'un ambitieux projet, mais la porte est ouverte à des recherches futures.

## APPENDICE A

### Fichiers internes de la banque de données de PLANAIR

Nous donnons ici le détail de l'organisation des fichiers internes de la banque de données de PLANAIR. Chaque fichier défini comporte un numéro interne (de 1 à 99), un type (entier, réel ou caractère), un nombre d'enregistrements et un nombre de mots par enregistrement. Nous indiquerons de plus à quelle(s) variable(s) des programmes correspondent les éléments de chaque fichier.

Plusieurs paramètres de compilation spécifiques à une application donnée sont définis. Parmi ceux-ci, mentionnons:

-BPW: nombre de bits par mots sur l'ordinateur utilisé

-MDAYS: nombre max. de jours dans une période de planification (MDAYS$\leq$366)

-NWORDS: nombre de mots nécessaires pour codifier les jours de la période sous forme de vecteurs de bits
NWORDS= INT( MDAYS + BPW - 1 /BPW)

F1: PARAMETRES GLOBAUX ET DE SCENARIO

TYPE: entier
# D'ENRG.: 2
MOTS/ENRG: 80
VARIABLE(S): premier enrg.:paramètres globaux
second enrg. :paramètres de scénario

Voici l'énoncé "blank common" qu'on retrouve au début de tous les modules pour y déclarer les paramètres globaux:

```
c
c
c     global parameters
c
      common ldi,ldo,lgi,lgo,ldai,ldao,lero,llio,lrep,lgraph,
     1       iphys(10),
     2       kmod,idev,ishort,lpsiz,ipge,idat,iusr,itpter,itpri,itpplo,
     3       ida31,nexmod,igcad,icpdat,iscom,imodl,lmodl,icge,iefb,ida,
     4       klu,ida,ida,iradi,ida45(6),
     5       nvtyp,nvclas,nmode,ngate,nact,nturn,ndays,nscen,ida59(2),
     6       ivtyp,nvclas,nmode,nwords,ida65(16)
```

Ces paramètres sont définis ainsi:

- dix premiers mots: adresses logiques des unités d'entrées/sorties

- dix mots suivants: adresses physiques des unités d'entrées/sorties

-KMOD: mode d'opération; 1=interactif, 2=en lot

-IDEV: paramètres du terminal graphique

-ISHORT:profil binaire ("bit pattern") servant à déterminer l'absence ou la présence de certaines parties du dialogue

-LPSIZ: nombre de lignes par page pour les rapports

-IPGE: numéro de la page courante du rapport

-IDAT: date courante

-IUSR: initiales de l'usager (4 car. , format A4)

-ITPTER:type du terminal courant

-ITPRI: type de l'imprimante courante

-ITPPLO:type du fichier graphique courant

-NEXMOD:numéro interne du prochain module à appeler à la sortie du module courant

-IGCMD: profil binaire déterminant les commandes graphiques disponibles

-ICPDAT:date de compilation du module courant

-ISCEN: scénario courant (numérotation interne)

-IMODL: module courant (entier à trois chiffres)

-LMODL: dernier module actif (entier à trois chiffres)

-ICGM: groupe de modules courant (utilisé par le menu principal)

-IMFB: booléen de modification de la base (0=o.k., 1=modification non complétée)

-KLU: abréviation pour l'unité de distance utilisée ("km" ou "mi")

-IRADI: dimension du rayon terrestre, en fonction de l'unité de distance utilisée

-MVTYP: nombre max de types de véhicules (BPW)

-MVCLAS:nombre max. de classes de véhicules

-MNODE: nombre max. de noeuds

-MGATE: nombre max. de points d'accès pour l'ensemble des noeuds

-MACT: nombre max. d'activités

-MTURN: nombre max. d'accrochages par scénario

-MDAYS: nombre max. de jours pour une période de planification (366)

-MSCEN: nombre max. de scénarios

-IVTYP: nombre de types de véhicules définis dans la banque ($0 \leq IVTYP \leq MVTYP$)

-NVCLAS:nombre de classes de véhicules définies dans la banque ($0 \leq NVCLAS \leq MVCLAS$)

-NNODE: nombre de noeuds définis dans la banque ($0 \leq NNODE \leq MNODE$)

-NWORDS: nombre de mots-machine nécessaires pour la codification de chaque horaire des activités et des accrochages

-IDM: espaces non utilisés dans le "blank common"

Voici l'énoncé "blank common" servant à déclarer des paramètres spécifiques à l'ensemble des données considérées.

```
c
c     scenario parameters
c
      common istats,iyear,imon,iday,jmon,jday,ndays,iendy,itrans,idm10,
     1       montab(14),idm25(6),
     3       iwday(7),idm38(3),
     4       nact,nturn,iafree,itfree,idm45(6),
     5       iwtab(&NWORDS&),
     6       idms(&30-NWORDS&)
```

Voici la définition de ces paramètres:

-ISTATS: profil binaire indiquant le statut du scénario

-IYEAR: année de début de la période considérée. Un signe négatif indique que la période se termine l'année suivante.

-IMON: entier donnant le mois de début de la période ($0<IMON\leq12$)

-IDAY: entier donnant le jour de début de la période ($0<IDAY\leq31$)

-JMON: entier donnant le mois de fin de la période ($0<JMON\leq12$)

-JDAY: entier donnant le jour de fin de la période ($0<JDAY\leq12$)

-NDAYS: nombre de jours dans la période considérée ($0\leq NDAYS\leq MDAYS$)

-IENDY: numéro du jour dans la période où a lieu un changement d'année ($1\leq IENDY\leq367$)

-NACT: nombre d'activités définies ($0\leq NACT\leq MACT$)

-NTURN: nombre d'accrochages définis ($0\leq NTURN\leq MTURN$)

-IAFREE: pointeur vers le premier emplacement libre pour les activités. Une valeur nulle indique qu'il n'y a plus d'espace. ($0\leq IAFREE\leq MACT$)

-ITFREE:pointeur vers le premier emplacement libre pour les accrochages. Une valeur nulle indique qu'il n'y a plus d'espace. ($0 \leq ITFREE \leq MTURN$)

F2: TITRE DU PROJET ET DU SCENARIO

TYPE: texte (A2)
# D'ENRG: 2
# MOTS/ENRG: 20
VARIABLE(S): IPTIT(10) , ISTIT(10)

Titres de 40 caractères définis par l'usager.

F3: TITRES DES ATTRIBUTS DONNES PAR L'USAGER

TYPE: reel
# D'ENRG: 4
# MOTS/ENRG: 6
VARIABLE(S): KDATA(6)

Pour chaque entité (noeuds, classes de véhicules, activités, accrochages), l'usager dispose de trois attributs numériques (réels). Pour chacun de ces attributs, un titre de huit lettres peut être défini. C'est ce titre qui apparaîtra dans les différents rapports et dialogues.

F4: PRIORITE DES JOURS DE LA PERIODE

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MDAYS
VARIABLE(S): IPRIOR(MDAYS)

Tableau utilisé lors de la conversion des vecteurs de bits.

F10: IDENTIFICATEURS DES NOEUDS

TYPE: texte (A4)
# D'ENRG: 1
# MOTS/ENRG: MNODE
VARIABLE(S): IDNODE(MNODE)

Liste des identificateurs des noeuds ordonnés alphabétiquement.

F11: COORDONNEES DES NOEUDS

TYPE: réel
# D'ENRG: 2
# MOTS/ENRG: MNODE
VARIABLE(S): RLAT(MNODE) , RLONG(MNODE)

F12: HEURE LOCALE DES NOEUDS

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MNODE
VARIABLE(S): LOHR(MNODE)

Nombre de minutes de déviation, par rapport au temps GMT, pour chaque noeud.

F13: PREMIERE ACTIVITE

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MNODE
VARIABLE(S): IFACT(MNODE)

Pointeurs sur la première activité des noeuds.

F14: POINTS D'ACCES AUX NOEUDS

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MNODE
VARIABLE(S): IGATE(MNODE)

F15: ATTRIBUTS DES NOEUDS

TYPE: réel
# D'ENRG: 1
# MOTS/ENRG: MNODE*3
VARIABLE(S): UNODE(MNODE*3)

Attributs définis par l'usager.

F16: PARAMETRES D'ACCES AUX NOEUDS

TYPE: entier
# D'ENRG: 2
# MOTS/ENRG: MGATE
VARIABLE(S): premier enrg:JVTYPE , second enrg:ITRAN

F18: MATRICE DES DISTANCES ENTRE LES NOEUDS

TYPE: réel
# D'ENRG: MNODE
# MOTS/ENRG: MNODE
VARIABLE(S): MDIST(MNODE,MNODE)

F30: IDENTIFICATEURS DES CLASSES DE VEHICULES

TYPE: texte (A4)
# D'ENRG: 1
# MOTS/ENRG: MVCLAS
VARIABLE(S): IVCLAS(MVCLAS)

F31: NOMBRE DE VEHICULES PAR CLASSE

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MVCLAS
VARIABLE(S): NUMVEH(MVCLAS)

F32: ATTRIBUTS DES CLASSES DE VEHICULES

TYPE: réel
# D'ENRG: 1
# MOTS/ENRG: MVCLAS*3
VARIABLE(S): ITVAT(MVCLAS*3)

Attributs définis par l'usager.

F33: TYPE DE VEHICULES AUQUEL LA CLASSE DE VEHICULES APPARTIENT

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MVCLAS
VARIABLE(S): KTYP(MVCLAS)

F50: TABLEAU ORDONNE DES IDENTIFICATEURS DES ACTIVITES

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MACT
VARIABLE(S): IORD(MACT)

F51: IDENTIFICATEURS DES ACTIVITES

TYPE: texte (A4)
# D'ENRG: 1
# MOTS/ENRG: MACT
VARIABLE(S): IDACT(MACT)

F52: DONNEES TEMPORELLES DES ACTIVITES

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MACT
VARIABLE(S): IATIME (un mot pour chaque activité)

Chaque élément du tableau IATIME contient trois valeurs comprimées:
IATIME = IBTIME (temps de départ), bits 21 à 32
IETIME (temps d'arrivée), bits 9 à 20
LAGA (durée en jours ), bits 1 à 8

F53: ORIGINE-DESTINATION DES ACTIVITES

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MACT
VARIABLE(S): IOD (un mot pour chaque activité)

IOD = INODE (noeud de départ), bits 17 à 32
JNODE (noeud d'arrivée), bits 1 à 16

F54: POINTEURS DES ACTIVITES PAR NOEUDS

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MACT
VARIABLE(S): NEXT (un mot pour chaque activité)

NEXT = INEXT (activité suivante à INODE), bits 17 à 32
JNEXT (activité suivante à JNODE), bits 1 à 16

F55: VEHICULES DES ACTIVITES

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MACT
VARIABLE(S): IAVEH(MACT)

IAVEH = IVEH (classe de véhicules au point de départ), bits 17 à 32
JVEH (classe de véhicules au point d'arrivée), bits 1 à 16

F56: ATTRIBUTS DES ACTIVITES

```
TYPE:          réel
# D'ENRG:      1
# MOTS/ENRG:   MACT*3
VARIABLE(S):   IAAT(MACT*3)
```

Attributs définis par l'usager

F58: JOURS EFFECTIFS DES ACTIVITES

```
TYPE:          entier (vecteur de bits)
# D'ENRG:      1
# MOTS/ENRG:   MACT*NWORDS
VARIABLE(S):   IADAYS(MACT*NWORDS)
```

F59: JOURS-INTERVALES DES ACTIVITES

```
TYPE:          entier
# D'ENRG:      1
# MOTS/ENRG:   MACT
VARIABLE(S):   INTERA(MACT)
```

```
INTERA =IFAD   (premier jour effectif), bits 23 à 32
        ILAD   (dernier jour effectif), bits 13 à 22
        NEVENT (cardinalité de IADAYS), bits  1 à 12
```

F60: POINTEURS VERS LES ACCROCHAGES

```
TYPE:          entier
# D'ENRG:      1
# MOTS/ENRG:   MACT
VARIABLE(S):   LINKT(MACT)
```

```
LINKT = ITURN (premier accrochage prédécesseur), bits 17 à 32
        JTURN (premier accrochage successeur  ), bits  1 à 16
```

F70: ACTIVITES RELIEES PAR LES ACCROCHAGES

```
TYPE:          entier
# D'ENRG:      1
# MOTS/ENRG:   MTURN
VARIABLE(S):   LINKA(MTURN)
```

```
LINKA = IENDA (activité se terminant), bits 17 à 32
        IBEGA (activite débutant    ), bits  1 à 16
```

F71: DUREE DES ACCROCHAGES

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MTURN
VARIABLE(S): ITLAG(MTURN)

F72: POINTEURS SUR LES LISTES D'ACCROCHAGES

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MTURN
VARIABLE(S): NEXTUR(MTURN)

NEXTUR = NEXEND (prochain accrochage prédécesseur), bits 17 à 32
NEXBEG (prochain accrochage successeur ), bits 1 à 16

F73: JOURS EFFECTIFS DES ACCROCHAGES

TYPE: entier (vecteur de bits)
# ENRG.: 1
# MOTS/ENRG: MTURN*NWORDS
VARIABLE(S): ITDAYS(MTURN*NWORDS)

F74: JOURS-INTERVALES DES ACCROCHAGES

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: MTURN
VARIABLE(S): INTERT(MTURN)

INTERT = IFTD (premier jour effectif), bits 23 à 32
ILTD (dernier jour effectif), bits 13 à 22
NTDAYS (cardinalité de ITDAYS), bits 1 à 12

F75: ATTRIBUTS DES ACCROCHAGES

TYPE: réel
# D'ENRG: 3
# MOTS/ENRG: MTURN
VARIABLE(S): ISTAT(MTURN)

F91: JOURNAL DE BORD

TYPE: entier+texte
# D'ENRG: 1
# MOTS/ENRG: MLOGB
VARIABLE(S): LOGB(MLOGB)

F92: LIGNES DE DEMARCATIONS

TYPE: réel
# D'ENRG: 9
# MOTS/ENRG: MDEML

F93: POINTEUR ET NOMBRE DE PARAMETRES POUR CHAQUE MODULE

TYPE: entier
# D'ENRG: 1
# MOTS/ENRG: 1000

F94: PARAMETRES DE DEFAUT POUR LES MODULES

TYPE: réel
# D'ENRG: 60
# MOTS/ENRG: 20
VARIABLE(S): PARM(20)

F95: FICHIER D'ESPACE TEMPORAIRE DE TRAVAIL

TYPE: réel
# D'ENRG: 2
# MOTS/ENRG: max(MACT*NWORDS , MTURN*NWORDS)+5

F96: FENETRES GRAPHIQUES PREDEFINIES

TYPE: réel
# D'ENRG: 27
# MOTS/ENRG: 9
VARIABLE(S): RSW(9)

F97: TABLE DES EQUIPEMENTS GRAPHIQUES

TYPE: entier
# D'ENRG: 25
# MOTS/ENRG: 60

## REFERENCES


ADLER, R.K. and P. RICHARDUS. Map Projections. Amsterdam, North-Holland, 1972. 174p.

BABIN, A., M. FLORIAN, L. JAMES-LEFEVBRE and H. SPIESS. EMME/2: User's Manual. Centre de recherche sur les transports, Université de Montréal, 1982 (and later updates). 500p.

BARNARD, D.R. "Aircraft Feasibility Model". AGIFORS 20th Annual Symposium Proceedings, Miami, AGIFORS, 1980. pp.549-564.

BIRD, C.J. and W.R. McCALLUM. "A Model for Allocation of Aircraft to Events in an Operational Schedule". AGIFORS 11th Annual Symposium Proceedings, Miami, AGIFORS, 1971. pp.93-105.

ETSCHMAIER, M.M. and D. MATHAISEL. "Aircraft Scheduling: the State of the Art". Presented at the Twenty-fourth Annual AGIFORS Symposium, 1984.

IBM Inc. Airline Schedule Planning in the 1980s. Feltham, IBM inc., 1980. 39p.

KAYUKAWA, K. "ARAJIN ( Aircraft Routing and Automatically Job Integrating at Narita Base Station ) SYSTEM". AGIFORS 23rd Annual Symposium Proceedings, Miami, Eastern airlines, 1983. pp.265-280.

MAILING, D.H. Coordinate Systems and Map Projection. London, George Philip, 1973. 255p.

RANDALL, D.S. and A.H. ROBINSON. Elements of Cartography. New-York, John Wiley, 1969. 415p.

SHNEIDERMAN, B. Software Psychology: Human Factors in Computer and Information Systems. Cambridge, Winthrop Publishers, 1980. 320p.

SPIESS, H. Contributions à la théorie et aux outils de planification des réseaux de transport urbain. Thèse de doctorat, Dép. d'informatique et de recherche opérationnelle, Université de Montréal, 1984. 296p.

USEROS, T. "Flight Schedule Mechanization Project: Frequencies Version System". AGIFORS 15th Annual Symposium Proceedings, Miami, AGIFORS, 1975. pp.364-414.

## REMERCIEMENTS

Je tiens à remercier:

Michael Florian, mon directeur de recherche. Sa grande expérience et l'appui soutenu qu'il m'a prêté ont contribué grandement à la réalisation de ce travail.

Heinz Spiess, mon co-directeur de recherche. J'ai vraiment apprécié ses conseils judicieux et ses critiques constructives, mais je lui suis, avant tout, redevable de l'acquisition d'une méthode de travail très "cartésienne".

La société Air Canada. L'intérêt manifesté pour ce projet et l'assistance technique apportée ont permis de faire le lien entre le développement théorique et l'application pratique.

Lucie Bibeau et Isabelle Constantin. Par leurs encouragements et leurs commentaires, elles m'ont apporté une aide précieuse dans la production de la version finale de ce mémoire.

Marie-France, ma compagne. Elle a su véritablement me seconder pendant tout ce temps.